FH Technikum Wien

MASTER THESIS

zur Erlangung des akademischen Grades
„Master of Science in Engineering“
im Studiengang Mechatronik / Robotik (Master)

RobotaS – Entwicklung eines mobilen Roboters zum Kleinwaren-Transport

ausgeführt von BSc Dimitrios Prodromou
1200 Wien, Leystraße 69/35-36

1. Begutachter: Dipl.-Ing.Dr.techn. Michael Zillich
2. Begutachter: Oberst dhmtD DI Bernhard Peschak

Wien, 12.08.2011

FACHHOCHSCHULE
TECHNIKUM WIEN

Kurzfassung

Diese Masterarbeit befasst sich mit der Entwicklung eines Prototypen zum Kleinwaren-Transport in Supermärkten. Dabei soll ein mobiler Roboter einer Person ab dem Betreten bis zum Verlassen des Supermarktes als Transporthilfe bereitstehen. Im Verlauf dieser Masterarbeit wird vor allem auf die verwendete Kinect Kamera eingegangen. Zusätzlich zum RGB-Bild wird durch diese ein Tiefenbild der Umgebung erstellt, welches in der späteren Berechnung der Position der Zielperson von großer Bedeutung ist. Basierend darauf werden die verwendete Vision Bibliothek OpenCv und das Robot Operating System, zur Detektion und Verfolgung der Zielperson und verschiedene Bildmanipulationsmethoden im 2D- und im 3D- Bildbereich betrachtet. Aus der Kombination der gewählten Methoden bilden sich schließlich die Version 1 (3D-Punktwolke und spezieller Template matcher) und die schnellere und verbesserte Version 2 (Tiefenbild und speziell angepasster Particle Filter) des Gesamtsystems. Des Weiteren wurden verschiedene drahtlose Systeme zur eindeutigen Identifizierung und Distanzberechnung der Zielperson überprüft. Dabei konnte das Bluetooth-System, neben ZigBee, RFID und WLAN, als einziges getestet werden, da es das kostengünstigste Produkt und sofort verfügbar war. Die durchgeführten Experimente ergaben allerdings keine ausreichend genaue Distanzangabe der Person, weswegen keines der drahtlosen Systeme in das Gesamtsystem implementiert wurde. In den abschließenden Kapiteln werden die beiden Versionen des Gesamtsystems, ihr kompletter Aufbau und die durchgeführten Experimente im Detail erörtert. Die Experimente mit beiden Systemen ergaben eine gute Detektion der Zielperson in verschiedenen Szenarien. Dabei wurde diese bei konstanten Lichtverhältnissen zwischen verschiedenen Objekten und anderen Personen mit einer hohen Übereinstimmung gefunden. Bei variablen Lichtverhältnissen traten hingegen verfälschte Farbinformationen auf, was zu Berechnungsfehlern und schließlich zur verfälschten Übereinstimmungsergebnissen führte. Zusammengefasst betrachtet wurde eine funktionierende, sehr gute Basis für weitere Projekte gelegt. Der Prototyp ist uneingeschränkt ausbaufähig und liefert schon in dieser frühen Phase ausgesprochen gute Ergebnisse.

Schlagwörter: OpenCv, Robot Operating System, Kinect, Einkaufsroboter, drahtlose Systeme

Abstract

This master thesis deals with the development of a prototype for small goods transport in supermarkets. The mobile robot should follow a person and carry her goods while being inside the supermarket. The mainly discussed hardware component is the used Kinect camera. In addition to the RGB image, a depth image of the environment is created, which is a necessary part in the position calculation of the target person. Based on this, the vision library OpenCv and the Robot Operating System, for detecting and tracking the target person, are presented. In addition to that, various image manipulation methods for 2D and 3D images are shown. From the combination of the chosen methods, the version 1 (3D point cloud and special template matcher) and the faster and improved version 2 (depth image and specially designed particle filter) of the overall system were developed. Furthermore, some wireless systems have been checked for clear identification and distance calculation of the target person. In addition to ZigBee, RFID and WLAN, only the Bluetooth system could be tested, since it was the least expensive product and also immediately available. The conducted experiments resulted in insufficient accurate distance information of the person. Therefore none of the wireless systems has been implemented into the overall system. In the concluding chapters, the two versions of the overall system, their complete construction and the performed experiments are discussed in detail. The experiments with both systems resulted in a good detection of the target person in different scenarios. It was found that, at constant light conditions, between different objects and other people a high degree of fidelity was given. For variable light conditions, however, distorted color information appeared, resulting in calculation errors and finally to wrong results. In summary, a very good basis for further projects was developed. The prototype is fully expandable and provides very good results at this early stage.

Keywords: OpenCv, Robot Operating System, Kinect, robotic shopping card, wireless systems

Eidesstattliche Erklärung

„Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.“

Ort, Datum	Unterschrift

Danksagung

Besonderer Dank geht an meine beiden Betreuer Dr.techn. Michael Zillich und Oberst dhmtD DI Bernhard Peschak für ihre nicht endende Unterstützung bei der Entwicklung des Prototypens und bei der Erstellung dieser Masterarbeit. Durch ihre Hilfe konnten viele Probleme und vermeidliche Sackgassen überwunden werden. Vor allem durch die kontinuierliche Unterstützung und Mitarbeit bei der Implementierung des von Dr.techn. Michael Zillich angepassten Particle Filters, wurde die Version 2 des Gesamtsystems erst ermöglicht.

Dem gesamten Team des ACIN-Institutes der Technischen Universität Wien, danke ich für ihr Feedback und Input bei auftretenden Problemen. Viele Teile dieser Arbeit wurden erst durch ihre Hilfe ermöglicht.

Ich danke auch Herrn Dr.techn. Markus Vincze für die Gelegenheit meine Masterarbeit am ACIN-Institut absolvieren zu können.

Auch meiner Lebenspartnerin Bettina Haller danke ich für ihre ständige Unterstützung und Hilfestellung an späten Abenden. Ohne ihr immer offenes Ohr für meinen Prototypen, wäre die Projektzeit ziemlich einsam geworden.

Zu guter Letzt möchte ich mich noch bei Herrn Dipl.-Ing. Mag. David Fischinger für die zahlreichen Tischfußballspiele bedanken. Gerade in Momenten in denen scheinbar nichts mehr zu funktionieren schien, konnte eine gute Partie Tischfußball oftmals die Lösung herbeiführen. Am Ende ist er immer noch der bessere Spieler.

Inhaltsverzeichnis

1 Einführung 1
1.1 Projektdefinition 1
1.2 Aufbau dieser Arbeit 2
1.3 Stand der Technik 6
1.3.1 Mobile Robot Platform 6
1.3.2 Multi-Sensor Human Tracking 6
1.3.3 Zusammenfassung der Ergebnisse 7

2 Überblick über die Basiskomponenten 8
2.1 Basis Software 8
2.1.1 Betriebssystem 8
2.1.2 Entwicklungssprache C++ 8
2.1.3 Entwicklungsumgebung 9
2.1.4 Robot Operating System 9
2.1.5 OpenCV 9
2.2 Basis Hardware 10
2.2.1 Roboter 10
2.2.2 Laptop 11
2.2.3 Kinect Kamera 11
2.2.4 Sensoren 12
2.2.5 Controller 15

3 2-Dimensionale Bildbearbeitung 16
3.1 Erweiterte Grundlagen 16
3.1.1 Exkurs RGB-Farbmodell 16
3.1.2 Exkurs HSV-Farbmodell 17
3.2 Betrachtete Methoden 18
3.2.1 Kantenerkennung mit verschiedenen Algorithmen 18
3.2.2 Konturen finden 19
3.2.3 Bewegungserkennung 20
3.2.4 Ellipsen finden 21
3.2.5 Regionen von Interesse 22
3.2.6 Template matching 23

3.2.7 Fast Template Matching 24
3.2.8 Farbhistogramme 25
3.2.9 Histogrammvergleiche 27
3.2.10 Color Histogram Matching 28
3.2.11 Flood Fill 29
3.2.12 Scale Invariant Feature Transform 30
3.3 Zusammenfassung der Ergebnisse 31

4 3-Dimensionale Bildbearbeitung 34
4.1 Erweiterte Grundlagen 34
4.2 Betrachtete Methoden 39
4.2.1 Die 3-Dimensionale Punktwolke 39
4.2.2 Filterungsmethoden der 3-Dimensionalen Punktwolke 40
4.2.3 Tiefenbild Filterung 42
4.3 Zusammenfassung der Ergebnisse 43

5 Particle Filter 45
5.1 Aufbau des Particle Filters 45
5.2 Fazit Particle Filter 47

6 Drahtlose Systeme 48
6.1 Identifikation mittels Radio Frequency Identification 49
6.1.1 Grundlegender Aufbau 50
6.1.2 Frequenzen und Reichweite 51
6.1.3 Passive und aktive Transponder 51
6.1.4 Fazit Radio Frequency Identification 52
6.2 Identifikation mittels Bluetooth 52
6.2.1 Grundlegender Aufbau eines Identifikationssystems 53
6.2.2 Versuche mit Bluetooth 54
6.2.3 Fazit 59
6.3 Identifikation mittels WLAN 59
6.4 Identifikation mittels ZigBee 60
6.4.1 Versuch mit ZigBee 61
6.4.2 Fazit 61
6.5 Vergleiche zwischen den Funksystemen 61
6.5.1 Vergleich zwischen Bluetooth und ZigBee 61

6.5.2 Vergleich zwischen Bluetooth und WLAN 62
6.5.3 Vorteile von Bluetooth gegenüber RFID 63
6.5.4 Nachteile von Bluetooth gegenüber RFID 63
6.6 Zusammenfassung der Ergebnisse 64

7 Robotersteuerung 65
7.1.1 Steuerung des Roboters 65
7.1.2 Vorderen Hindernissen erkennen und ausweichen 66
7.1.3 Hinteren Hindernissen erkennen und ausweichen 67

8 Umsetzung des Prototypen 69
8.1 Version 1 – 3D-Punktwolke + FastTemplateMatcher 69
8.1.1 Startpunkt (main) 70
8.1.2 Bildverarbeitung (updateVisualisation) 70
8.1.3 Initialisierungsphase (initPhase) 70
8.1.4 Punktwolkenfilterung (filterPointCloud) 71
8.1.5 Region-von-Interesse (setRegionOfInterest) 71
8.1.6 Initialisierungsbilder (initScreenshot) 71
8.1.7 Programmablaufphase (runPhase) 72
8.1.8 Umwandlung von 3D- zu 2D-Bildern (convert3Dto2D) 73
8.1.9 Fast Template Matching (fastTemplateMatcher) 75
8.1.10 Distanzberechnung (calculateMeanDistance) 75
8.1.11 Robotersteuerung (robotControlAndPublish) 76
8.1.12 Sicherheitsschalter (safetyButton) 76
8.2 Version 2 – Tiefenbild + Particle Filter 77
8.2.1 Ablauf des Programms im System 78
8.2.2 Startpunkt (main) 78
8.2.3 RGB- und Tiefenbild (get_rgb_from_kinect- und get_depth_from_kinect) 79
8.2.4 Bildbearbeitung (imageProcessing) 79
8.2.5 Sicherheitsschalter (safetyButton) 80
8.2.6 Initialisierungsphase (init_phase) 80
8.2.7 Filterfunktion (filter_rgb) 80
8.2.8 Region-von-Interesse Berechnung (calculateInitROI) 81
8.2.9 Initialisierungshistogramm (takeInitHistogramm) 81
8.2.10 Programmablaufphase (run_phase) 82

8.2.11 Particle Filter (particleFilter) 82
8.2.12 Robotersteuerung (robotControlAndPublish)....83
8.2.13 Abstands- und Rotationsregler (distancePIDController- und rotationPIDController) 84
8.3 Ablauf des Programms für den User 85
8.3.1 Person stellt sich vor die Kamera 85
8.3.2 Person steuert den Roboter mittels Controller 85
8.3.3 Person bewegt sich 85
8.3.4 Person beendet das Programm....86
8.4 Versuche und Evaluierungen der beiden Systeme 86
8.4.1 Version 1 des Gesamtsystems....86
8.4.2 Version 2 des Gesamtsystems....87
9 Zusammenfassung und Ausblick.... 91
Abbildungsverzeichnis.... 97
Tabellenverzeichnis.... 99
Literaturverzeichnis 100

1 Einführung

Stationäre Roboter, vor allem in der Industrie, vereinfachen die täglich anfallende Arbeit, wie Schweißen, Bohren oder Entgraten schon seit Jahrzehnten. In ihrem Bereich sind sie teilweise nicht mehr wegzudenken, besonders in Arbeitsumgebungen, welche für den Menschen gefährlich werden können.
In Zukunft werden diese Roboter immer mobiler werden, um die Menschen noch besser bei ihrer Arbeit und vor allem im täglichen Leben zu unterstützen. Durch den Einsatz von mobilen Robotern im Staubsauger- und im Rasenmäherbereich wurden schon die ersten Schritte in diese Richtung getätigt. Diese simplen Gehilfen verrichten ihre Arbeit überwiegend in Bereichen, welche für einen Roboter eine relativ geringe Intelligenz und Selbstentscheidung erfordern. Meistens ohne „Augen“, bewegen sie sich nahezu blind in ihrer Umgebung und reagieren auf geringe Kollisionen oder Annäherungen an Wände oder Hindernisse. Was für den simplen Haushalt vollkommen ausreichend ist, wäre für den Einkauf in einem Supermarkt allerdings zu gering. Hierbei spielen je nach Anforderung viel mehr Faktoren eine Rolle, als sie der simple Aufbau eines Staubsaugroboters bewältigt.

1.1 Projektdefinition

Der Grundgedanke dieser Masterarbeit ist es, einen Prototyp eines vollautomatischen Einkaufswagens zu entwickeln. Dieser soll einer Person ab dem Betreten des Supermarktes als Assistent bereitstehen und ihr beim Tragen der Einkäufe helfen, bis diese den Supermarkt wieder verlässt. Durch den behindertengerechten Aufbau der meisten Supermärkte ist es ihm somit möglich, der Person nahezu überallhin zu folgen. Die Testumgebung für dieses Projekt befindet sich dabei im ACIN-Institut der Technischen Universität Wien.

Aufgrund dieser Aufgabenstellung ergeben sich nun folgende primären und optionalen Ziele:

Primäre Ziele:

- Der Roboter erkennt die Zielperson in der Initialisierungsphase und folgt ihr durch den Raum
- Hindernisse wie Regale, Mülleimer etc. werden nicht als Ziele erkannt
- Andere Personen, welche sich in der Umgebung der Zielperson befinden oder zwischen Roboter und Zielperson durchgehen, werden ebenfalls nicht in als Ziele betrachtet
- Sollte der Roboter die Zielperson verlieren, findet er sie mittels der vorhandenen Kamera wieder

Optionale Ziele:

- Verschiedene Funktechnologien sollen miteinander verglichen und nach eingehender Recherche, eine Implementierung in das Gesamtsystem in Betracht gezogen werden
- Roboternavigation um Hindernissen auszuweichen und zur Kollisionsvermeidung mittels des Laser-Sensors
- Wiederfindung des Ziels in einer größeren Umgebung
- Objekte welche sich im Weg befinden sollen automatisch umrundet und das Ziel wieder in den Fokus gebracht werden

Vorteile:

- Personen brauchen ihren Einkaufswagen nicht mehr selber schieben
- das Potential der Weiterentwicklung des Systems ist überaus groß (Scannen der Waren, Wunschliste von Stammkunden, etc.)
- Für körperlich beeinträchtigte Menschen (dazu können auch ältere Menschen zählen) stellt das System eine erhebliche Erleichterung der Erledigung eines Einkaufes dar

Herausforderungen:

- Variable Lichtbedingungen
- Geschwindigkeiten des Roboters
- Personen mit ähnlichen Kleidungen

1.2 Aufbau dieser Arbeit

Die ständige und rasante Entwicklung von neuen Technologien hat auch in dieser Masterarbeit Einzug gefunden. Dies zeichnet sich dadurch aus, dass während der Projektlaufzeit, zwei Programme mit unterschiedlichen Funktionsweisen entwickelt wurden. Dazu mussten im Vorfeld verschiedene Experimente durchgeführt und miteinander verglichen werden. Die nachfolgenden Kapitel sollen daher eine Einsicht in den kompletten Aufbau und die Entwicklung des Prototypens geben. Außerdem soll eine kurze Übersicht im weiteren Verlauf die grundlegenden Kapitel zusammenfassen:

Basis Software

Besonders der Softwarebereich spielt in diesem Projekt eine große Rolle. Ein Programm dieser Komplexität benötigt eine gewisse Basis Software, welche bereits erfolgreich in anderen Projekten eingesetzt werden konnte. Mit dem Fokus auf Stabilität, Komfort und Effektivität, werden die benutzten Programme ausgesucht und vorgestellt. Aufgrund der geringen Zeit, wird dabei von aufwendigen vergleichen verschiedener Hersteller abgesehen.

Um einem mobilen Roboter zu ermöglichen einer Person zu folgen, benötigt er ein System welches mit ihm kommuniziert und ihm z.B. Bewegungsbefehle schickt. So ein System ist

das hier verwendete Robot Operating System (ROS). Neben den Treibern zur Steuerung des Roboters sind auch Treiber zur Kommunikation mit verschiedenen Kameratypen vorhanden und können über Pakete leicht nachinstalliert werden. Die Informationen in den empfangenen Kamerabildern müssen schließlich entsprechend verarbeitet werden. Für diese Aufgabe eignet sich die Vision Bibliothek OpenCv am besten dafür. Sie ist quelloffen und beinhaltet alle Funktionen zur Erkennung und Verfolgung einer Person in einem Bild.
Idealerweise wird nun eine Programmiersprache gewählt, welche von beiden Programmen verwendet wird und noch dazu sehr weit verbreitet ist. Dadurch kann eine Recherche nach bestimmten Funktionen oder eine Fehlersuche im Internet besonders vereinfacht und beschleunigt werden. Die Programmiersprache, welche die hier genannten Kriterien am besten erfüllt, ist dabei C++. Außerdem wurde diese im Zuge des Studiums ausführlich in anderen Projekten angewendet.

Basis Hardware

Die Basis Hardware setzt sich in diesem Falle aus Komponenten zusammen, welche entweder extra dafür ausgewählt wurden oder bereits im ACIN-Institut der Technischen Universität Wien vorhanden waren. Aus diesem Grund wird auch auf einen mobilen Roboter der Firma Pioneer zurückgegriffen. Dieser erfüllt weitgehend alle Projektanforderungen und kann sofort eingesetzt werden. Des Weiteren ist er mit verschiedenen Sensoren ausgestattet, welche zur Navigation- und/oder zur Kollisionsvermeidung und -detektion eingesetzt werden können.
Um die Distanz einer Person über bildgebende Verfahren zu ermitteln, wird eine spezielle Kamera, oder eine Kombination aus Kameras (Stereokamera-Systeme) benötigt. Auch hierbei wurde auf einen Vergleich verschiedener Kameramodelle verzichtet, da die verwendete Kinect-Kamera bei einem günstigen Anschaffungspreis, überaus gute Bilder ermöglicht. Die Verwendung diese Kamera im Bereich der Robotik ist mittlerweile sehr weit fortgeschritten und aus diesem Grund auch immer besser dokumentiert.

2-Dimensionale Bildverarbeitung

Bei der gestellten Aufgabenstellung wird vor allem eine stabile Identifikation der Zielperson benötigt. Hierbei sollen verschiedene Merkmale einbezogen werden, welche eine Person von einer anderen unterscheidbar macht. Das Problem, welches sich nun stellt, ist die Filterung von wichtigen und unwichtigen Bildinformationen. Außerdem müssen die gewählten Merkmale im Bild definiert und wiedergefunden werden können. Die Vision Bibliothek OpenCv beinhaltet genau für diese Anwendungsfälle verschiedene Funktionen, welche in Kombination, die geforderten Kriterien entsprechen können. Die geeigneten Funktionen gilt es allerdings im Vorfeld herauszufiltern, da nicht jede Funktion für diese Aufgabestellung geeignet ist. Aus diesem Grund werden einige Funktionen im Detail betrachtet und ihre Implementierungsmöglichkeiten in das Gesamtsystem überprüft.

3-Dimensionale Bildverarbeitung

Wie bereits erwähnt, benötigt das System eine Möglichkeit die Distanz zu der Zielperson zu ermitteln. Die verwendete Kinect-Kamera liefert zwar die benötigten Tiefeninformationen, allerdings müssen diese noch nach bestimmten Kriterien gefiltert werden. ROS bietet hierfür die Point Cloud Library (PCL) an, welche verschiedene Methoden zur Filterung und Bearbeitung von 3D-Punktwolken ermöglicht. Hiermit können wiederrum, wichtige von unwichtigen Informationen getrennt werden. Neben der 3D-Punktwolke, kann die Distanz auch über das vorhandene Tiefenbild ermittelt werden. Diese Möglichkeit wurde erst im Laufe des Projektes verwendet und führte schließlich zur Version 2 des Gesamtsystems.

Particle Filter

Die Position einer Person in einem Supermarkt wird sich mit großer Wahrscheinlichkeit ständig ändern. Diese kann über die bildgebenden Verfahren und einer Wahrscheinlichkeitsberechnung durch den Particle Filter ermittelt werden. Aus der Kombination der Ergebnisse aus den 2D und 3D-Bildverarbeitungsfunktionen überprüft er laufend die Wahrscheinlichkeit, ob die aktuell erfasste Person, auch die Zielperson ist.

Drahtlose Systeme

Der Einsatz von Identifikationschips würde einen Roboter genau zu einer Zielperson für den Aufenthalt im Supermarkt binden. Mithilfe dieser zusätzlichen Identifizierung, wird die Verfolgung der Zielperson nochmal verbessert. Verschiedene Technologien wurden nun theoretisch analysiert und miteinander verglichen. Aufgrund der teilweise enormen Anschaffungskosten, wurde nur das bereits in Laptop und Mobiltelefon implementierte Bluetooth zu Testzwecken herangezogen.

Robotersteuerung

Ein Roboter kann über einen Controller oder über ein Programm, welches Bewegungssignale schickt, gesteuert werden. Über ROS kann die Robotersteuerung einfach realisiert werden, da er schon den Treiber für den Pioneer Roboter implementiert hat und dieser relativ einfach in das Gesamtsystem integriert werden kann. Die Befehle müssen vorher entsprechend aufbereitet werden, damit der Roboter diese auch korrekt interpretieren kann.

Dabei kann es durchaus dazu kommen, dass sich Hindernisse im Pfad des Roboters befinden, welche durch die Kamera nicht erfasst werden. Herfür werden die montierten Sensoren getestet, um sie eventuell auch in das Gesamtsystem zu integrieren.

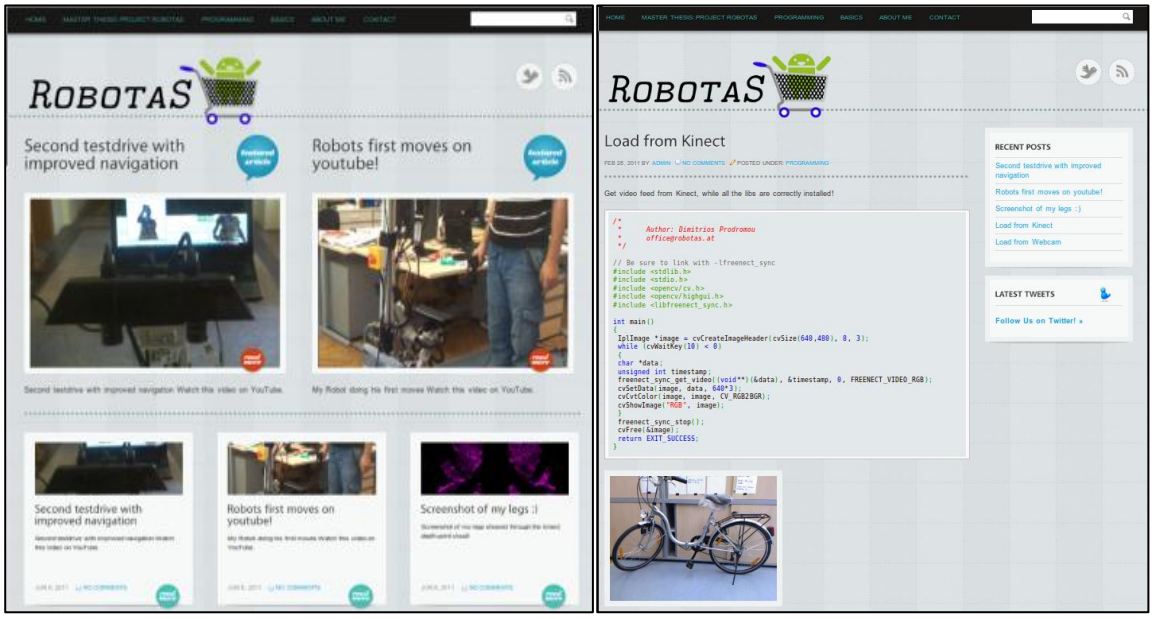

Abbildung 1: Offizielle Website des RobotaS-Projekts

Während der Projektlaufzeit wurde über eine Webseite (Abbildung 1) im Internet über den aktuellen Verlauf des Projektes berichtet. Interessierten Personen und Firmen erhielten somit auf der Seite http://www.robotas.at eine Möglichkeit, sich über das Projekt zu Informieren und Fragen zur Implementierung oder zur eingesetzten Hardware zu stellen.

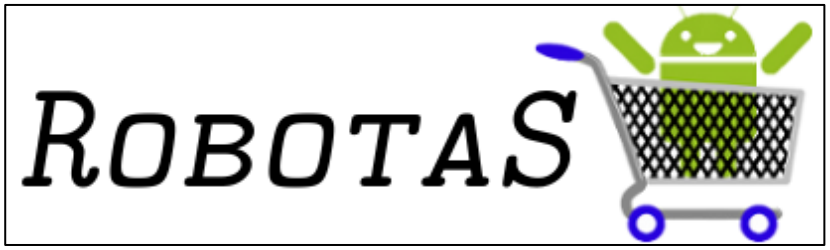

Abbildung 2: Offizielles Logo des RobotaS-Projekts

Außerdem wurde ein offizielles Projektlogo entworfen, welches den Wiedererkennungswert des Projektes steigern soll (Abbildung 2).

Sicherheitsaspekte

In einer Umgebung mit vielen Personen sollte auch der Sicherheitsaspekt im Umgang mit einem Roboter nicht vernachlässigt werden. So muss z.B. gewährleistet sein, dass eine Bewegung unterbunden wird, wenn sich ein Hindernis im Pfad des Roboters befindet. Bei einer Kollision müssen entsprechende Maßnahmen durchgeführt werden.
Obwohl die Kollisionsvermeidung mithilfe des installierten Lasers kein primäres Ziel dieser Arbeit ist, wird darauf in Kapitel 7.1.2 näher eingegangen und einige Versuche mit dem Hokuyo Laser durchgeführt.

1.3 Stand der Technik

Dieses Kapitel soll einen kurzen Überblick über zwei Projekte geben, welche sich auch mit dem Problem der Personenverfolgung beschäftigt haben. Zuerst wird die Mobile Robot Platform der Firma Cytron Technologies betrachtet, welche das gleiche Ziel eines automatisierten Einkaufwagens verfolgt. Das zweite Projekt, Multi-Sensor Human Tracking, wurde an der University of New South Wales in Australien von vier Studenten entwickelt, welche sich mit dem Erkennen, Verfolgen und Wiederfinden einer Zielperson zwischen diversen anderen Personen beschäftigt.

1.3.1 Mobile Robot Platform

Das Malaysische System mit dem Namen Mobile Robot Platform der Firma Cytron Technologies dient nach eigenen Angaben als Schul- und Forschungsplattform. Über dieses System gibt es nur geringe Dokumentationen, außer zwei Beiträgen auf der Video Plattform Youtube. In diesen werden die einzelnen Komponenten des Systems veranschaulicht und der Roboter kann in Aktion beobachtet werden. Dieses System besteht aus einem eigenständig zusammengebauten Roboter mit zwei bürstenlosen 30W Motoren zum Antrieb der beiden Räder. Die Umdrehungen dieser Räder werden mit Hilfe zweier Rotations-Encoder erfasst. Außerdem befindet sich eine steuerbare Pan-Tilt Webcam auf der oberen Plattform. Ein Hokuyo Laser Range Finder erfasst und verfolgt die Person, welche sich im Aktivierungsaugenblick vor dem Roboter befindet. Verschiedene Microcontroller übernehmen die Steuerung des Systems. Die Spannungsversorgung übernehmen zwei (12V, 7.2Ah) SLA Akkumulator Batterien.

In den einzigen vorhandenen Quellen, den Youtube-Videos, scheint das System überaus robust seine Zielperson zu verfolgen. Da es allerdings keine wissenschaftlichen Dokumentationen des Aufbaus oder der Ergebnisse gibt, kann nicht von einem 100% autonomen System ausgegangen werden. In dem Video, welches den Roboter in realer Umgebung zeigt, fehlt außerdem die Kamera. Somit wird die Person nur durch den Laser-Range-Sensor detektiert und verfolgt. Dies macht das System instabil gegenüber Personen, welche sich zwischen Roboter und Zielperson befinden. Es kann davon ausgegangen werden, dass der Sensor den einem Bein ähnelnden Konturen, welche sich vor ihm befinden, folgt. Hilfe bei der Steuerung des Roboters mittels eines Controllers ist dabei nicht auszuschließen. Dieses Projekt sollte deswegen eher als gute Marketingkampagne der Firma Cytron Technologies gewertet werden. (Cytron Technologies, 2009)

1.3.2 Multi-Sensor Human Tracking

Das Multi-Sensor Human Tracking Projekt wurde von vier Studenten der School of Computer Science and Engineering, der University of New South Wales in Australien

entwickelt. Ziel des Projektes war es, ein System zu konzipieren, welches einer Zielperson folgen kann, auch wenn sich andere bewegliche Objekte in der unmittelbaren Umgebung des Roboters befinden. Dazu wurde vor allem die Methode der Beinerkennung mittels eines Sick-Laserscanners verwendet. Verliert der Roboter sein Ziel, schaltet sich die vorhandene Kamera ein und sucht nach allen oberkörperähnlichen Objekten in der Umgebung. Wird ein Objekt gefunden, wird dieses mithilfe eines vorhandenen Initialisierungsbildes auf seine Richtigkeit hin überprüft. Im Projektbericht werden allerdings auch die Schwierigkeiten bei der Wiederfindung der Zielperson im Kameramodus hingewiesen. Dabei wurde aus Zeitmangel keine stabile Methode implementiert, einem Hindernis, welches sich genau zwischen Zielperson und Roboter befindet, zuverlässig auszuweichen. Da die Kamera die Person eindeutig identifiziert hat, erkennt der Laser-Scanner in diesem Fall das Hindernis als korrektes Ziel an.
In der finalen Testreihe wurden einige wichtige Funktionen dieses Projektes, wie z.B. die automatische Kollisionsvermeidung, nicht implementiert. Dennoch wurden gute Ergebnisse erzielt und das Projektziel erreicht. (University of New South Wales, 2008)

1.3.3 Zusammenfassung der Ergebnisse

Die Betrachtung des Standes der Technik hat vor allem verdeutlicht, dass auf diesem Gebiet zwar gearbeitet wird, allerdings noch keine nennenswerten Erfolge verzeichnet werden konnten. In manchen Bereichen konnten diese Prototypen zwar eingesetzt werden, es fehlt aber weiterhin die Möglichkeit es für kommerzielle Produkte anwendbar zu machen. Dieses Projekt soll ein weiter Schritt in dieser Richtung sein, wobei aus den Erkenntnissen und Erfahrungen der anderen Projekte profitiert wird.

2 Überblick über die Basiskomponenten

Zu Beginn des Projektes, wurden verschiedene Komponenten getestet und miteinander verglichen. Somit konnte zu diesem Zeitpunkt gewährleistet werden, dass diese auch den Projektzielen förderlich sind. In den nachfolgenden Kapitel 2.1 und 2.2 wird der Fokus daher auf die grundlegend benötigte Software und die benutzte Hardware gelegt.

2.1 Basis Software

Die verwendete Basis-Software wurde durch Tests, oder die Erfahrung im Umgang mit ihr ausgewählt. Aufgrund der Tatsache, dass sich einige Softwarepakete als Internationaler und Institutsweiter Standard abzeichnen und die Zeit zum Testen limitiert war, wurde auf eine Vorstellung und ein Vergleich diverser Systeme verzichtet. Die folgenden Kapitel geben unter anderem einen Überblick über das verwendete Betriebssystem Ubuntu (2.1.1) und die OpenSource Bibliothek OpenCv (2.1.5), sowie das Robot-Operating-System (ROS, 2.1.4).

2.1.1 Betriebssystem

Als Betriebssystem wurde das lizenzfreie und kostenlos erhältliche Ubuntu 10.10 verwendet, welches von der Ubuntu Homepage heruntergeladen und auf nahezu jedem PC oder Laptop installiert werden kann. Vor allem das im weiteren Verlauf dieser Arbeit erläuterte Robot Operating System (ROS, 2.1.4) zur Steuerung des Roboters und der Kinect-Kamera benötigt dieses Linux basierte System. Die Installation unter Windows ist durch ein Zusatztool namens Wubi möglich und bedarf keiner zusätzlichen administrativen Veränderung der Festplatten Partitionen. Sollte Ubuntu nicht mehr benötigt werden, kann es normal deinstalliert und dadurch wieder entfernt werden. Der Speicher steht anschließend wieder dem System zur Verfügung. Im Zuge der ständigen Entwicklung von Ubuntu ist mittlerweile die Version 11.40 erschienen, welche aber aufgrund von möglichen Inkompatibilitäten mit der installierten Software oder dem eigentlichen Programm selber nicht getestet wurde.

2.1.2 Entwicklungssprache C++

Die Suche nach einer Entwicklungssprache wurde dahingehend vereinfacht, da die hierbei vorgestellten Softwarepakete, alle mit der gleichen Entwicklungssprache programmiert werden können. Aus diesem Grund wurde hierfür C++ als die gemeinsame Programmiersprache gewählt. Dieser Code kann in jedem Editor geschrieben und mit den erforderlichen Befehlen und Compilern auch kompiliert und ausgeführt werden. Da diese Entwicklungssprache seit einigen Jahrzehnten verwendet und teilweise allgemeiner Standard geworden ist, wird nicht näher darauf eingegangen. Eine detaillierte Anleitung der Benutzung, der Sprache, oder die Verwendung diverse Compiler kann in entsprechenden Fachliteraturen nachgelesen werden.

2.1.3 Entwicklungsumgebung

Um die Arbeit in größeren Projekten zu erleichtern und die Funktionalität zu erweitern, wird auf die Entwicklungsumgebung Eclipse zurückgegriffen. Alternative Entwicklungsumgebungen wie NetBeans oder KDevelop wurden zusätzlich getestet, konnten aber dabei nicht mit dem Komfort und den Funktionalitäten von Eclipse mithalten. Mit dieser können umfangreiche Projekte erstellt und verwaltet werden. Die gesamten Bibliotheken werden dabei in das System eingebunden und stehen jedem Programm zur Verfügung. Außerdem beinhaltet es einen Modus, welcher dem Benutzer Codevorschläge präsentiert und einen Debug-Modus, welcher bei einer Fehlersuche behilflich ist. Dadurch kann ein Code schneller entwickelt und die Suche nach genauen Codephrasen oder Fehlern beschleunigt werden.

2.1.4 Robot Operating System

Um den verwendeten Pioneer-Roboter steuern zu können wird das quelloffene und frei erhältliche Robot Operating System (ROS) verwendet. ROS wurde entwickelt um jeden Roboter mit einer gemeinsamen Software steuern zu können. Somit kann ein Roboterentwickler auf ein komplett funktionierendes System zurückgreifen, welches nicht extra für den einzelnen Roboter entwickelt werden muss. Das ROS-System ist modular erweiterbar und wird ständig mit Hilfe einer sehr großen Community auf dem neuesten Stand der Technik gehalten. Neben der Robotersteuerung können hiermit auch die für dieses Projekt benötigten Bereiche der Bildverarbeitung und Funksysteme in ROS programmiert und benutzt werden. Somit wird eine vollständige Bearbeitung in einem kompletten System ermöglicht. Die betrachteten Methoden und weitere Informationen zur Programmierung von ROS finden sich im Kapitel 4 der 3-Dimensionale Bildbearbeitung und im Kapitel 7 der Robotersteuerung. Einige Programmierbeispiele wurden außerdem im Anhang beigelegt.

2.1.5 OpenCV

Um die Informationen der Kamerabilder be- und verarbeiten zu können bedarf es einer Vielzahl von Algorithmen und Funktionen. Erst durch die Kombination dieser, kann eine Person in einem Bild zunächst einmal erkannt und anschließend auch verfolgt werden. Aus diesem Grund wird die quelloffene Vision-Bibliothek OpenCv verwendet. Diese beinhaltet eine Reihe von Algorithmen z.B. zur Kantenerkennung mit dem Canny-, oder zur Gesichtserkennung mit dem HaarCascade- Algorithmus. Diese wurde ursprünglich von der Firma Intel veröffentlicht und wird nun von der Firma Willow-Garage gepflegt und weiterentwickelt. Dabei beinhaltet sie so gut wie den gleichen Funktionsumfang, wie der Vision-Bereich des kommerziell erhältlichen Programms MatLab. Der Vorteil hierbei liegt vor allem in der großen Community, welche sich schon mit einer Vielzahl von unterschiedlichen Problemen befasst hat und somit eine schnellere Lösung herbeiführen

kann. Die betrachteten Methoden rund um die OpenCv-Bibliothek werden im Kapitel 3 der 2-Dimensionale Bildbearbeitung ausführlich erläutert.

2.2 Basis Hardware

Ergänzend zum vorherigen Kapitel 2.1, wird nun der Fokus auf die Basis-Hardware gelegt. Unter anderem wird der benutzte Roboter (2.2.1) und die Kinect-Kamera (2.2.3) vorgestellt, sowie weitere Hardwaretechnische Komponenten, wie der benötigte Controller (2.2.5) oder die Sensoren zur Navigation- und Kollisionsdetektion, wie der Laser Sensor von Hokuyo (2.2.4) und die installierten Bumper (2.2.4).

2.2.1 Roboter

Wie bereits erläutert, verfügt das ACIN-Institut der Technischen Universität Wien über einen mobilen Roboter der Firma Pioneer (Modell P3-DX, Abbildung 3), welcher modular erweitert werden kann. Dieser besitzt bereits vormontierte Sensoren, um einfache Tests durchführen zu können. Zu dieser Hardware gehören unter anderem:

- acht Ultraschall- Sensoren, welche den vorderen Bereich des Roboters abtasten
- ein Hokuyo-Laserscanner, welcher eine Ebene in einer Höhe von ca. 30cm und einer Fläche von 240° abtastet
- und zwei DC-Motoren, welche zwei 19 cm Reifen antreiben

Die maximale Geschwindigkeit des Roboters beträgt 1,6 m/s und das maximale Tragegewicht ist auf 23 kg beschränkt. Des Weiteren sind drei 12 V/7,6 Ah Blei-Akkumulatoren zur Stromversorgung des gesamten Systems verbaut. Diese Stromversorgung wird auch für die Kinect-Kamera und die Sensoren verwendet.

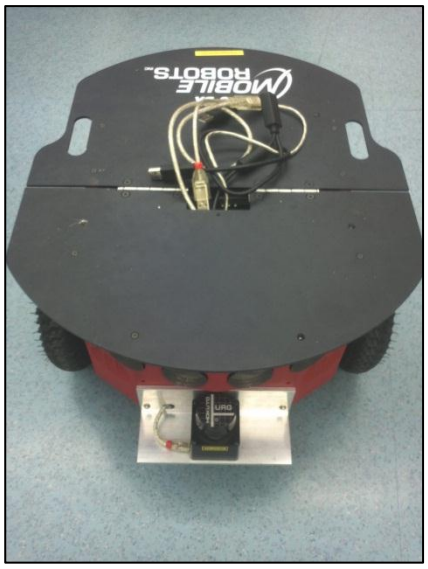

Abbildung 3: Verwendeter Pioneer Roboter

Dieser Roboter eignet sich ideal für die gestellte Aufgabe, da er bereits ROS-Treiber dafür gibt. Darüber hinaus verfügt er bereits über die eventuell benötigten Sensoren für die Kollisionsvermeidung.

2.2.2 Laptop

Der Roboter wird über ein Linux basierendes System bedient (siehe Kapitel 2.1.1), welches ständig mit ihm kommunizieren muss. Dazu wurde ein Laptop verwendet, welcher über einen i5-DualCore Prozessor mit 2,4Ghz Rechenleistung und einer ATI-5560 Grafikarte mit 512Mb Grafikkartenchip verfügt. Auf diesem wurde die verwendete Basis-Software installiert um das Roboter-System aufzubauen.

2.2.3 Kinect Kamera

Eine der essentiellsten Hardwarekomponenten in diesem Projekt, findet sich sicherlich in der Wahl der Kamera. Diese liefert die Bilder zur Personendetektion und -verfolgung. Idealerweise können durch diese auch die Tiefeninformationen der Umgebung erfasst werden.
Im Oktober 2010 wurde eine neue Kamera mit dem Namen „Kinect“, durch die Firmen Microsoft und PrimeSense vorgestellt. Bei dieser Farb- und Tiefenbildkamera für die Spielkonsole Xbox360 handelt es sich um eine neuartige Technik, Spiele rein durch die Bewegung des Körpers und der Hände zu steuern. Ein physischer Controller, wie er bislang üblich war, wird nun nicht mehr benötigt. Sie wird vor den Spielern positioniert und ist nach einer kurzen Eingewöhnungsphase bereits einsatzbereit.

Da diese Kinect Kamera (Abbildung 4) bereits in einigen vorherigen Projekten im ACIN-Institut erfolgreich eingesetzt wurde, konnte auf einen Vergleich verschiedener Kameramodelle verzichtet werden. Sie ist mittlerweile sehr gut dokumentiert und konnte in vielen Projekten effektiv eingesetzt werden. Durch den einfachen Aufbau und die hohe Qualität der gelieferten Tiefendaten, wurde diese relativ schnell auch für den Robotikbereich interessant. Vor allem der Preis, welcher momentan bei 100€ liegt, ist im Vergleich zu Kameras mit dem gleichen Funktionsumfang sehr gering. Ein weiterer Vorteil ist die fortschreitende Integration in das ROS-System.

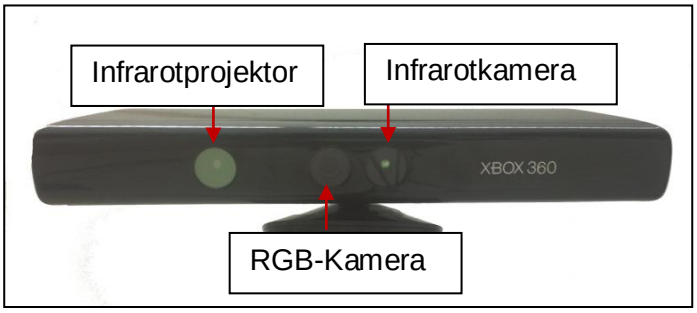

Abbildung 4: Die Kinect Kamera mit ihrem kompletten Aufbau

Aufbau der Kinect Kamera

Diese spezielle Kamera besteht aus einer RGB-Kamera, einem Infrarot-Projektor und einer Infrarot Kamera. Diese in Kombination ermöglichen es ein 3-dimensionales Bild in Farbe und in Tiefeninformationen zu generieren.

Die RGB-Kamera hat eine Auflösung von 640x480 Pixel und kann ein Farbbild in einem Format von 640x480 (32-Bit Farbtiefe) mit 30 Bilder/s wiedergeben. Dadurch können die zusätzlichen Live-Bilder für die spätere Erkennung und Verfolgung der Zielperson genutzt werden.

Um die Tiefeninformationen zu erhalten wird die Infrarot-Kamera in Kombination mit dem Infrarot-Projektor betrieben. Dabei wird ein Infrarot Muster durch den Projektor ausgestrahlt, welches mit dem menschlichen Auge nicht wahrgenommen werden kann. Die Infrarot-Kamera kann nun diese Punkte im Raum erkennen und sich durch die Zuhilfenahme der einfachen Triangulation die Entfernung zu dem Punkt im Raum berechnen (siehe Kapitel 4.1). Die Vor- und Nachteile der Kinect Kamera sind in Tabelle 1 ersichtlich.

Vorteil	Nachteile
- Kosten von 100€ - Große Community - bereits eine Vielzahl von durchgeführten Projekten - schnelle Implementierung	- Tiefenbilder von max. 3-4m - Treiber teilweise instabil - geringere Auflösungen als Hochauflösende Industriekameras

Tabelle 1: Vor- und Nachteile der Kinect Kamera

2.2.4 Sensoren

Bei den vorgestellten Sensoren, handelt es sich um teilweise bereits installierte Komponenten, welche den Funktionsumfang des Roboters nochmals zusätzlich erhöhen. Während der Projektlaufzeit wurde der Roboter gewechselt, weswegen die Bumper in der

finalen Version nicht mehr zur Verfügung standen. Nichtsdestotrotz sind sie ein wichtiger Sicherheitsfaktor und können ohne weiteres auch am neuen Roboter nachgerüstet werden, weswegen die Erklärung dazu, nach wie vor Gültigkeit besitzt.

Hokuyo Laser Ranger

Dieser Laserscanner (Abbildung 5) befindet sich vorne am Roboter und dient in erster Linie dazu, Hindernisse und andere Objekte zu detektieren. Er befindet sich auf einer Höhe von ca. 30 cm und kann mit einer Reichweite von ca. 20 mm bis 4 m die Umgebung in einem Scanradius von 240° überprüfen. Tabelle 2 fasst dabei die Vor- und Nachteile des Laserscanners zusammen.

Abbildung 5: Ansichten des Hokuyo Laser Scanner

Vorteil	Nachteile
- Sehr großer Suchradius - Große Distanzen erreichbar - Feine Auflösungen auch in großer Entfernung	- Kosten von über 2000€ - Detektion nur in einer Ebene

Tabelle 2: Vor- und Nachteile des Hokuyo Lasersensors

Die durchgeführten Versuche mit dem Hokuyo Lasersensor werden in Kapitel 7.1.2 präsentiert.

Bumper

Die Modular erweiterbaren Bumper (Abbildung 6) an der Vorder- und Rückseite dienen dem Sicherheitsaspekt als letzte Instanz zur Kollisionsdetektion, wenn bereits eine Kollision erfolgte. Eine Vermeidung dieser kann durch die Bumper nicht gewährleistet werden. Sie bestehen lediglich aus einer Aufprall- oder Pufferfläche, welche bei Berührung einen eingebauten Kipptaster umlegt. Somit kann eine Kollision robust als 0 oder 1 erkannt werden. Wird eine Detektion der Aufprallintensität benötigt, können erweiterte

Tastsensoren die Messungen durchführen. In Tabelle 3 werden nun außerdem die Vor- und Nachteile der Bumper zusammengefasst.

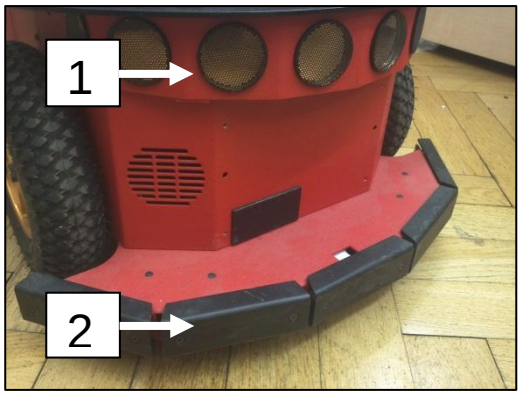

Abbildung 6: Sensoraufbau des alten Roboters
1. Ultraschall Sensoren, 2. Bumper

Vorteil	Nachteile
- Je nach Ausführung relativ günstig - Detektieren in einfacher Weise eine Kollision - Programmieraufwand ist äußerst gering (0-keine Kollision, 1-Kollision)	- Können Kollisionen nicht vermeiden - Bei einer Kollision kann bereits Schaden verursacht worden sein

Tabelle 3: Vor- und Nachteile der Bumper

Ultraschall Sensoren

Diese Sensoren werden in diesem Projekt nicht verwendet, weshalb die Vor- und Nachteile nur kurz erwähnt werden sollen (Tabelle 4). (nach Hesse, Schnell 2010, S. 58)

Vorteil	Nachteile
- berührungsloser Einsatz, somit kein Verschleiß - besonders zur Füll- und Grenzstandmessung geeignet - anders als Lasersensoren, können sie Glas detektieren	- Totzeit (aufgrund des Ausschwingens des Wandlers) nach dem Senden eines Ultraschallimpulses - Je nach Bauart (ältere Modelle), hörbares Klicken des Sensors - Streuung bei Reflexionen

Tabelle 4: Vor- und Nachteile der Ultraschallsensoren

2.2.5 Controller

Bei dem hierfür verwendeten Controller (Abbildung 7) handelt es sich um einen PS3 Wireless Controller mit Empfänger der Spielkonsole Playstation 3. Dieser ist ursprünglich dazu gedacht, Figuren in diversen Spielen zu steuern. Durch die Integration der Treiber in ROS, kann er hierbei auch für verschiedene roboterbezogene Anwendungen benutzt werden. Über die 12 verfügbaren Tasten können Funktionen abgerufen oder, wie z.B. ein Not-Aus-Schalter, gestartet werden. Mit dem vorhandenen Steuerkreuz und den zwei Joysticks können z.B. die Bewegungen eines Roboters gesteuert werden. Die Kommunikation zwischen Controller und Computer geschieht über einen beiliegenden Funkempfänger, welcher die Informationen verwaltet.

Abbildung 7: Bild vom verwendeten Controller

3 2-Dimensionale Bildbearbeitung

Die größte Herausforderung in diesem Projekt, ist die Erkennung und Verfolgung der Zielperson. Die Kinect Kamera liefert hierfür in erster Linie das aktuelle Bild der Umgebung mit allen seinen Inhalten, wie Personen, Objekten etc. Nun gilt es, diese Inhalte in wichtig und nicht-wichtig zu klassifizieren, um nur die Zielperson herauszufiltern. Um dies zu erreichen sind verschiedene Algorithmen erforderlich, welche in den nachfolgenden Beispielen erläutert und im Abschluss dieses Kapitels verglichen werden sollen. Jeder Algorithmus ist für seinen Bereich ideal geeignet und somit gilt es, die bestmögliche Kombination aus diesen zu ermitteln. Um die Programmierung dieser zu vereinfachen, wird die quelloffene OpenSource Vision Bibliothek OpenCv benutzt, da alle benötigten Algorithmen bereits integriert sind. Kapitel 3.2 dient dabei zur Veranschaulichung aller betrachteten Methoden, welche im anschließenden Kapitel 3.3 miteinander verglichen und die am besten geeigneten für die weitere Bearbeitung ausgewählt werden.

3.1 Erweiterte Grundlagen

Ein kurzer Exkurs im Bereich der RGB- und HSV-Farbmodelle dient zum besseren Verständnis im Bereich der Farbmanipulation und -bearbeitung.

3.1.1 Exkurs RGB-Farbmodell

Die meisten elektronischen bildgebenden Geräte wie Monitore, Scanner oder digitale Kameras, arbeiten mit genau diesem grundlegenden Farbmodell. Der Name RGB, setzt sich dabei aus den drei Grundfarben Rot, Grün und Blau zusammen und erklärt somit schon das Farbentstehungssystem. Werden diese drei Grundfarben vermischt, oder miteinander addiert, so können alle anderen Farben dadurch erzeugt werden. Eine einfache Weise, dies zu veranschaulichen, wäre ein Aufbau mit drei Diaprojektoren mit jeweils einer Grundfarbe. Bleiben alle drei aus, so ergibt sich die Farbe Schwarz. Wird immer nur einer eingeschaltet, so erzeugt er für sich seine Farbe. Werden nun zwei gleichzeitig auf die gleiche Fläche projiziert, so ergibt sich eine Vermischung dieser Farben und somit eine neue Mischfarbe. Aus der Farbe Grün und Rot ergibt sich z.B. die Farbe Gelb. Werden nun alle drei Projektoren angeschaltet, so ergibt die Vermischung oder Addition aller drei Farben die Farbe Weiß. Abbildung 8 zeigt dabei ein Bild in RGB-Farben. (Maschke, 2004, S. 41ff)

Abbildung 8: Ansicht in RGB-Farbmodell

3.1.2 Exkurs HSV-Farbmodell

Bei diesem Farbmodell handelt es sich um die Komponenten Farbton (Hue), Sättigung (Saturation) und Wert (Value). Anders als beim RGB-Farbmodell wird hierbei nicht ein Mischprodukt aus den Grundfarben Rot, Grün und Blau definiert, sondern eine Angabe eines genauen Farbtons mit einer bestimmten Sättigung und Intensität. Dies ermöglicht es, eine Farbe intuitiver auszuwählen, weshalb es in manchen Bildbearbeitungsprogrammen wie z.B. GIMP, standardmäßig als Farbwähler aktiviert ist. Dieses Farbmodell wird überwiegend als ein auf der Spitze stehender Kegel dargestellt. Ein Farbkreis von 0°-360° bildet dabei die kreisförmige Deckfläche mit den gesättigten Farbtönen (Hue). Eine Annäherung zur Kreismitte führt zu einer Vermischung mit der Farbe Weiß, was in einer Verblassung und schlussendlich zu einem Übergang ins reine Weiß führt. Durch diese Abnahme der Farbintensität wird der Wert für die Sättigung (Saturation) von 0-100 gewählt. Die Helligkeits-, oder Intensitätswerte (Value) von 0 bis 100 werden auf der vorhandenen senkrechten Mittelachse entsprechend der Lichtmenge, welche eine Farbe ausstrahlt, gebildet. Abbildung 9 zeigt dabei ein Bild in HSV-Farben. (Hornung, 2009, S. 260)

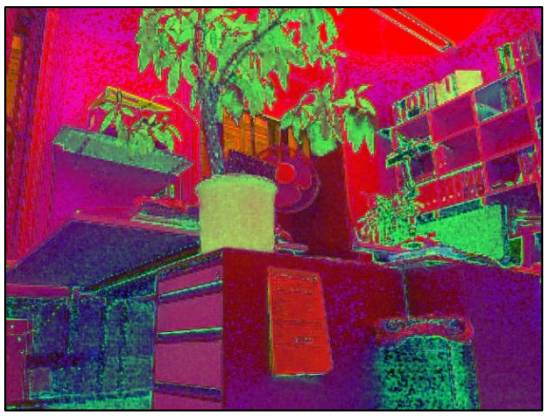

Abbildung 9: Ansicht in HSV-Farbmodell

Durch diesen klaren Aufbau des HSV-Farbmodells, kann eine eindeutige Trennung zwischen Farbinformationen (Farbton und Sättigung) und den Helligkeitsinformationen (Farbwert) erzielt werden. Wird hingegen in einem RGB-Farbmodell die Sättigung verändert, so ändern sich dementsprechend alle drei Farbwerte der drei Grundfarben. (Hornung, 2009, S. 260)

3.2 Betrachtete Methoden

Im weiteren Verlauf werden einige Beispiele und Funktionsweisen der für dieses Projekt relevanten Bildbearbeitung und -manipulation im Detail erläutert.

3.2.1 Kantenerkennung mit verschiedenen Algorithmen

Eine Überlegung eine Person in einem Bild zu identifizieren, wäre sie über ihre Form zu detektieren. Um diese zu erfassen, müssten ihre Kanten ermittelt werden können. Einer der grundlegenden Algorithmen zur Kantenerkennung in der Bildbearbeitung ist der Canny-Algorithmus. Dieses überaus schnelle und stabile Verfahren lässt sich relativ einfach in ein System integrieren. Das Ziel dieses Algorithmus ist es, die markanten Kanten in einem Bild hervorzuheben. Dazu können verschiedene Parameter das Endergebnis beeinflussen und dadurch z.B. nur die intensivsten Kanten hervorgehoben werden. Das eigentliche Bild wird dazu im Vorfeld in ein Graustufenbild umgewandelt und anschließend auf diesem der Canny-Algorithmus angewandt. Um diesen Prozess noch zu beschleunigen und die Kantenerkennung zu verbessern, können im Vorfeld Kantenoperatoren auf das Bild angewendet werden. Diese Algorithmen sind z.B. Difference of Gaussian oder Sobel, welche mehr oder weniger geeigneter sind, für die entsprechende Aufgabe. Der Canny-Algorithmus liefert basierend auf einem z.B. Sobel-Bild ein literarisiertes Kantenbild.

Wie auf den Bildern in Abbildung 10 zu erkennen ist, haben die verschiedenen Algorithmen eine größere oder kleinere Erfolgsquote bei der Detektion der Kanten. Die Parametereinstellungen dieser haben allerdings auch einen großen Einfluss auf das Ergebnis, weswegen bei diesen Beispielen nur die Standartwerte verwendet wurden. Beim Canny-Algorithmus werden vor allem feinere Kanten, detaillierter dargestellt. Hierbei kann nicht unterschieden werden, ob diese Kante nun markant, oder eine möglicherweise unwichtige ist. Auch die Ergebnisse der Kantenoperatoren Difference of Gaussian und Sobel können betrachtet werden. Dabei werden vor allem die äußeren Kanten sehr gut dargestellt und können eher von den inneren, evtl. nicht benötigten, unterschieden werden.

Abbildung 10: Kantendetektoren im Überblick
oben-links: Originalbild, oben-rechts: Canny-Algorithmus
unten-links: Difference of Gaussian, unten-rechts: Sobel-Algorithmus

Die Kantenerkennung ist in speziellen Anwendungsfällen überaus geeignet ein Objekt wiederzufinden. Des Weiteren verwenden einige Funktionen diese Algorithmen, als Ausgangsbasis für ihre eigenen Bearbeitungsschritte. Nichtsdestotrotz findet diese Funktion im Gesamtsystem keine Verwendung, da die Kanten einer Person zu sehr variieren, als dass eine stabile Personenverfolgung ermöglicht werden könnte.

3.2.2 Konturen finden

Soll sich die Detektion der Zielperson nur auf die markantesten Kanten konzentrieren, kann hierfür die FindContour-Funktion eingesetzt werden. Diese berechnet Konturen aus binären Bildern. Dabei können diese auf zwei Arten zur der Funktion zur Verfügung gestellt werden. Einerseits kann sie Bilder verwenden, welche bereits mit dem Canny-Algorithmus bearbeitet wurden und somit die Kantenpixel zur Verfügung stehen. Andererseits durch Bilder, welche mit der OpenCv Threshold-Funktion bearbeitet wurden, in welchen die Kanten die Grenze zwischen positiven und negativen Regionen angeben. (Bradski, Kaehler, 2008, S.234)
Das Ziel dieser Funktion ist es, vor allem die besonders markanten Kanten zu erkennen und diese graphisch darzustellen (Abbildung 11). Da diese Funktion auf einem Kantenerkennungsalgorithmus basiert, unterliegt sie den gleichen Identifikationsproblemen, wie bereits im Kapitel 3.2.1 erläutert wurde. Die Zuordnung der Kanten zu einer Person kann auch hierbei nicht stabil und sicher ermöglicht werden, weswegen diese Funktion nicht in das Gesamtsystem implementiert werden kann.

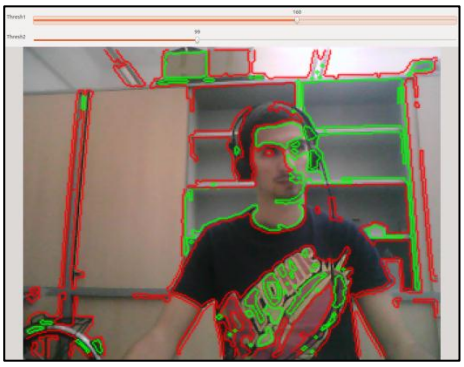

Abbildung 11: FindContour-Algorithmus auf ein Bild angewendet

3.2.3 Bewegungserkennung

Um eine Person in einem Bild zu finden ist es nötig, sie vom Hintergrund unterscheiden zu können. Eine Möglichkeit dies zu tun ist es, die Bewegung zu detektieren und das bewegte Objekt, in diesem Fall die Person selber, grafisch hervorzuheben. Das Problem, welches sich hierbei ergibt liegt darin, dass sich die Person immer bewegen muss, um auch lokalisiert zu werden. Auch sollte die Kamera statisch sein und sich somit nicht selbst bewegen, da sonst das Ergebnis erheblich verschlechtert wird. Bei dieser Methode werden die Pixel in einem Bild mit denen im nächsten Bild verglichen. Haben diese sich geändert, wurde eine Bewegung durchgeführt. Da sich der Hintergrund bei einem fix montierten Kamera-System idealerweise nicht ändert, entsteht auch keine Bewegung dieser Pixel und der Hintergrund kann dadurch ausgeblendet werden. Abbildung 12 zeigt dabei das Ergebnis der Bewegungserkennung. Wie im rechten Bild zu sehen ist, wurde die Bewegung der Person farblich hervorgehoben.

Abbildung 12: Bewegungserkennung mit Hintergrundentfernung

Wie bereits erwähnt, benötigt diese Funktion eine statische Kamera, welche sich selber nicht bewegt. Da sich der Roboter, auf welchem die verwendete Kinect- Kamera installiert

ist, ständig bewegt, kann diese Funktion nicht mehr zwischen Vorder- und Hintergrund unterscheiden und die Ergebnisse werden unbrauchbar. Für Kamerasysteme zur Personenzählung ist sie allerdings durchaus einsetzbar, wenn die Kamera an der Decke oder Wand montiert ist.

3.2.4 Ellipsen finden

Ist es aufgrund der Montage der Kamera nicht mehr möglich die gesamte Person als solche zu erkennen, müssen andere Verfahren eingesetzt werden, um z.B. Beine in einem Bild zu erkennen und wiederzufinden. Der FitEllipse-Algorithmus findet dabei die größten ellipsenförmigen Formen in einem Bild. Wird dieser auf einem Bild angewendet, welches nur aus den markantesten Kanten besteht (siehe Kapitel 3.2.1), kann eine gefundene Ellipse mit einer vorhandenen Ellipsenvorlage verglichen werden. Dadurch kann sich der Template matcher auf einen bestimmten kleinen Bereich fokussieren, statt im gesamten Bild zu suchen. Dies beschleunigt nochmal zusätzlich das Ergebnis.

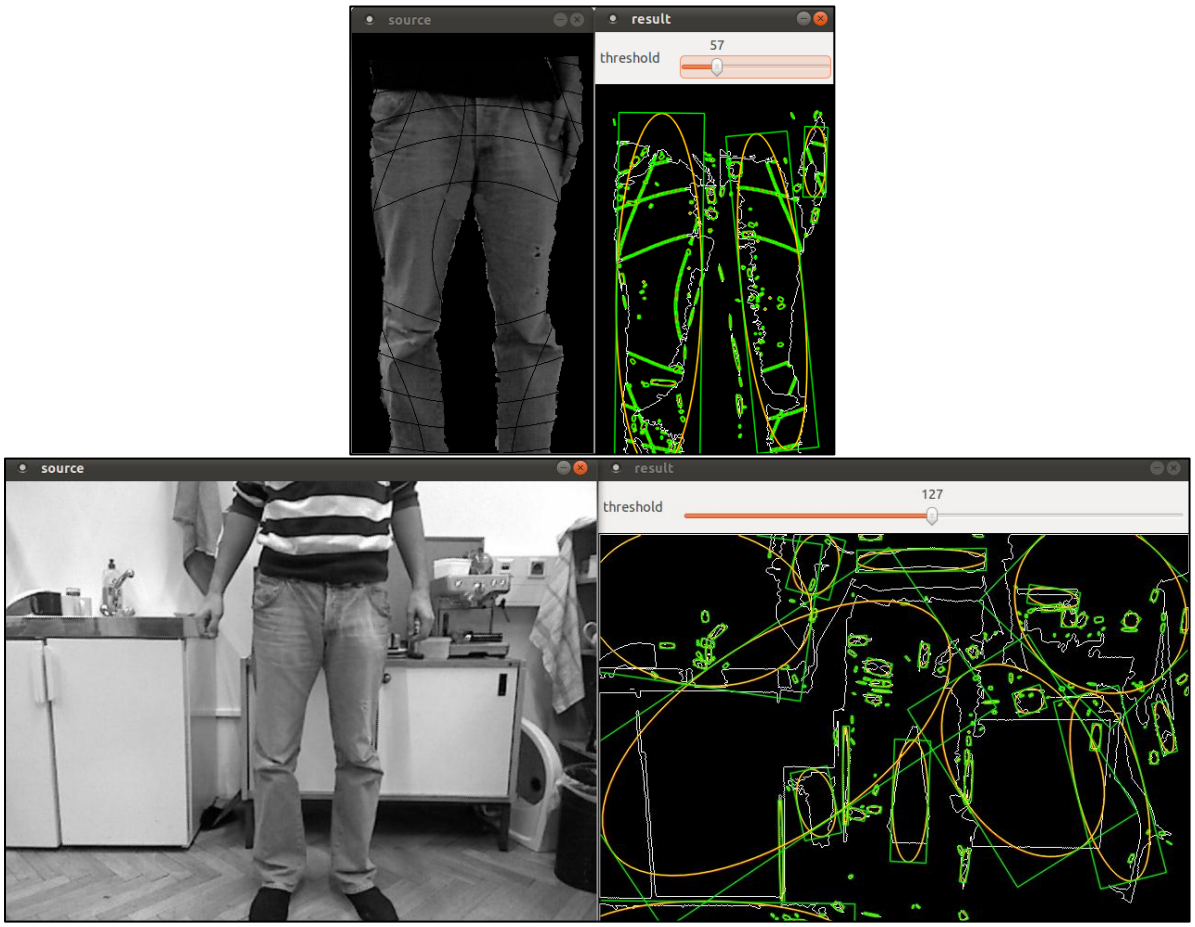

Abbildung 13: Detektion der Beine mit dem FitEllipsen-Algorithmus
(oben mit und unten ohne Segmentierung)

Allerdings ist der FitEllipse-Algorithmus überaus fehleranfällig, bei einer Rotation und teilweise Verdeckung des Suchobjektes im Bild. Er funktioniert nur in bestimmten Spezialfällen. Eine Person, welche sich bewegt, wird nie genauso dastehen wie zum Zeitpunkt der Initialisierung, bei der die Vorlage erstellt wurde. Des Weiteren verändert sich die Form der Beine während der Bewegung, weswegen auch nicht nach vordefinierten Ellipsen mit bestimmter Größe, im Bild gesucht werden kann. Personen mit Mänteln oder Röcken erschweren die robuste Detektion nochmals erheblich.

In diesem Test wurde versucht, die Beine einer Person in einem vorher segmentierten und in einem unsegmentierten Bild zu erkennen. Abbildung 13-oben zeigt zwei relativ gute, große Ellipsen, welche im segmentierten Bild gefunden wurden. Diese Information kann jetzt einer anderen Funktion übergeben und analysiert werden. Wie im unsegmentierten Bild (Abbildung 13-unten) deutlich zu erkennen ist, sucht der FitEllipse-Algorithmus lediglich nach den größten auffindbaren leeren Flächen im Bild, was die großen Ellipsen im Bereich der Schränke und der Wand erklärt. Somit ist eine Detektion der Beine nur in speziellen Fällen und nicht für eine reale Umgebung geeignet.
Wie bereits in den vorherigen Funktionen erläutert, ist auch diese nur in speziellen Anwendungsfällen einsetzbar. Vor allem die Form der Beine führen zu undefinierten Ergebnissen, welche eine stabile Detektion und vor allem Zuordnung zu einer Zielperson mit dieser Funktion unausführbar macht.

3.2.5 Regionen von Interesse

Die bereits erwähnten Verfahren können mit geringem Aufwand in ein System integriert werden. Teilweise haben sie aber den Nachteil, dass eine hohe Rechenleistung bei einem Livebild erforderlich ist, was unweigerlich zu einer langsamen Detektion führt. Dies geschieht dadurch, wenn z.B. eine gesuchte Vorlage im gesamten Bild gesucht wird. Eine Methode, dies zu vereinfachen, ist es, nur auf eine bestimmte Region zu schauen und diese dann näher zu untersuchen. Dadurch kann die Rechenzeit enorm verkürzt werden. Dieser Bereich wird Region-of-Interest (ROI, zu Deutsch: Region von Interesse) genannt und muss vorher definiert werden.

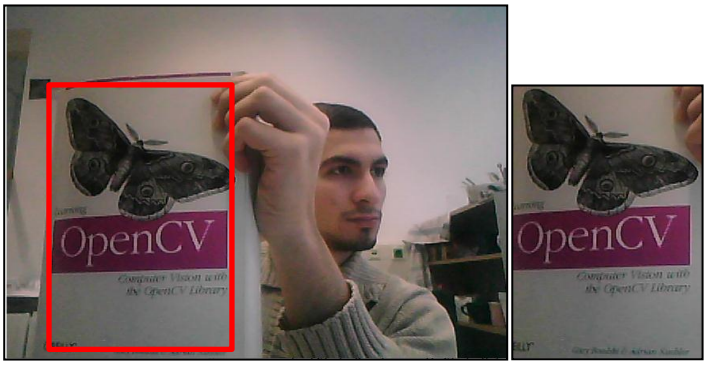

Abbildung 14: Originalbild mit markiertem (links) und ausgeschnittenem (rechts) Region-of-Interest

Ausgehend von einem definierten Kasten, wird das Bild nun auf den ROI reduziert. Die OpenCv-Funktion ermöglicht es diesen zu definieren. Dabei wird jeglicher Inhalt, welcher keine Relevanz für die weitere Bearbeitung hat, weggeschnitten und steht im weiteren Verlauf auch nicht mehr zur Verfügung. Abbildung 14-links zeigt dabei den Bereich, welcher für den ROI definiert wurde, wohingegen Abbildung 14-rechts das fertige Ergebnis der ROI-Operation anzeigt.
Durch die resultierende Zeitersparnis, ist diese Funktion in vielerlei Hinsicht einsetzbar. Besonders in zeitkritischen Anwendungen kann sie schnell eingebaut werden. In der Version 1 des Gesamtsystems findet sie Verwendung in der Erstellung der Initialisierungsbilder 8.1.3 und im FastTemplateMatcher 8.1.9. Hierbei wurde vor allem der Geschwindigkeitszuwachs im FastTemplateMatcher deutlich, da die gesuchte Vorlage nicht mehr über das gesamte Bild gesucht wird. In der Version 2 des Gesamtsystems wird die ROI bei der Erstellung des Initialisierungshistogramms 8.2.9 und beim Particle Filter 8.2.11 eingesetzt.

3.2.6 Template matching

Eine Methode zwei Bilder miteinander zu vergleichen, ist das Template matching. Beim diesem, wird dem System eine vorher definierte Vorlage zur Verfügung gestellt. Auf der Suche nach starken Übereinstimmungen, wird diese Vorlage über das aktuelle Bild geschoben und Pixel-für-Pixel verglichen. Hierbei besteht das Problem, dass die Vorlage durch Verschiebung eines Suchfensters über das ganze Bild gesucht wird. Bei einem Video-Stream mit 30 Bildern/s kann dies sehr rechenaufwendig werden. Außerdem können verdrehte oder verkleinerte Objekte nicht mehr eindeutig gefunden werden. In Abbildung 15-links ist die zu suchende Vorlage sichtbar. Schließlich wird diese in Abbildung 15-rechts im aktuellen Bild gefunden.

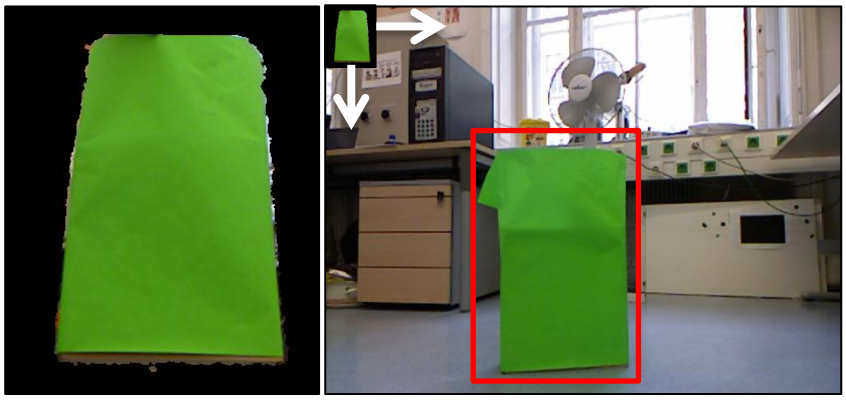

Abbildung 15: Vorlage (links) und gefundenes Übereinstimmung (Kasten rechts)

Der Template matcher ist relativ gut für Anwendungsfälle, welche nicht in Echtzeit ablaufen. Für die Implementierung in das Gesamtsystem ist er allerdings zu langsam.

3.2.7 Fast Template Matching

Die Tests mit dem normalen Template matcher (siehe Kapitel 3.2.6) haben vor allem gezeigt, wie rechenintensiv ein direkter Vergleich beider Bilder ist. Beim FastTemplateMatcher (ursprünglich Fast Match Template genannt, wurde dieser aufgrund der eigenen Modifikationen im weiteren Verlauf nur noch FastTemplateMatcher genannt) von Tristen Georgiou (Georgiou, 2010) wird neben dem normalen Template matching noch eine Downsampling (Verringerung der Bildpunkte) des Bildes vorgenommen, um die Berechnung zu beschleunigen. Die Geschwindigkeit der normalen Template matcher-Funktion ist direkt abhängig von der Größe des vorliegenden Bildes, weswegen ein Downsampling eine signifikante Steigerung in der Berechnung bedeutet. Georgiou errechnete darüber hinaus, dass bei einer konstanten Größe des Suchbildes die Anzahl der Berechnungen bei einem halb so großen Vorlagenbild ca. 50 Millionen (bei Verwendung der CV_CCORR_NORMED-Methode) beträgt. Bei größeren oder kleineren Vorlagenbildern verringert sich dementsprechend die Anzahl der Kalkulationen. Beträgt das Such- und Vorlagenbild nur noch ¼ der Originalgröße, so sind es nur noch 2 Millionen Kalkulationen. Somit kann folgender Algorithmus überlegt werden:

1. Ist das Suchbild 320*240 Pixel groß so setze i = 1, ist es größer, setze i = 2. Bei einem zu kleinen Bild, führt ein weiteres Downsampling, unweigerlich zu Verlusten in der Bildqualität. Bei einem zweifachen Downsampling verringert sich die Bildgröße schon um das 4-fache und die Berechnung wird beschleunigt. (Georgiou führte weitere Tests durch und kam zu dem Resultat, dass weitere Downsamplingvorgänge die Geschwindigkeit nicht maßgeblich steigern).
2. Nutze die cvPyrDown Funktion um das Bild i-fach downzusamplen.
3. Wende die cvMatchTemplate-Funktion auf das downgesampelte Bild an.
4. Finde die besten numLocation Treffer im Suchbild wieder.

5. Führe eine Koordinatentransformation der Bildpunkte zurück ins Originalbild durch.
6. Wende die cvMatchTemplate-Funktion auf die original großen Bilder an, aber setzte den Region-of-Interest im Suchbild, auf die gleiche Größe des Vorlagenbildes und zentriere es auf die numLocation Position.
7. Wiederhole Punkt 6., solange bis ein brauchbarer Wert erzielt wurde.

Dadurch wird das Programm um ein Vielfaches effektiver und schneller als das alleinige Template matching. Ein Vergleich beider Varianten ergab bei einem 640x480 Pixel großem Bild eine Verbesserung der Suche vom Sekunden- in den Millisekundenbereich, bei gleichbleibender Qualität des Ergebnisses. Wird nun ausschließlich in einem Region-of-Interest gesucht, beschleunigt sich die Suche noch einmal enorm.
Dabei kann zusätzlich noch eine Gewichtung nach der prozentualen Wahrscheinlichkeit des richtigen Treffers vorgenommen werden. Dies lässt einen gewissen Spielraum in der richtigen Detektion zu. Auch ist er nicht so störanfällig gegenüber einer Veränderung in Größe oder Rotation des zu suchenden Objektes.
Anders als der reine Template matcher, ist die Geschwindigkeit bei der Verfolgung einer Person um ein vielfaches schneller. Aus diesem Grund wird der FastTemplateMatcher auch in die Version 1 des Gesamtsystems eingebunden und ist die grundlegende Methode bei der Detektion und Verfolgung der Zielperson.

3.2.8 Farbhistogramme

Eine weitere Überlegung eine Zielperson in einem Bild zu detektieren, wäre sie über die Farbe ihrer Kleidung eindeutig zu identifizieren. Dazu müsste diese Information eingespeichert und speziell aufbereitet werden, um sie anderen Funktionen leichter zugänglich zu machen. Eine Möglichkeit dies umzusetzen, wäre durch die Erstellung eines Histogramms. Dieses gibt Informationen über die Anzahl der Pixel mit einem bestimmten Wert. Wird für jeden Kanal (R, G, B) jeweils ein Histogramm generiert, so ergibt sich ein Farbhistogramm, welches eine genauere Aussage über die Verteilung der Farbwerte in einem Bild ermöglicht. Dabei kann die Darstellung von diesem tabellarisch oder in graphischer Form erfolgen. Die x-Achse beinhaltet alle Farbwerte von 0 bis 255, wohingegen auf der y-Achse die Anzahl der Bildpunkte aufgetragen sind. Abbildung 16 zeigt dabei ein RGB-Bild mit den dazugehörigen Histogrammen für jeden einzelnen Kanal.

Abbildung 16: Farbhistogramm aus einem RGB-Bild

Die wichtigen Informationen über die Farbverteilung werden aus dem Funktionsverlauf gewonnen. Jedes Maximum in einem Farbhistogramm repräsentiert einen besonders intensiven Farbwert. Allerdings können aus einem Histogramm keine Informationen über die örtliche Anordnung gewonnen werden, weswegen ein Bild mit zwei zusammenhängenden dunklen und hellen Bereichen dasselbe Histogramm mit zwei Maxima besitzen kann wie eines, welches statistisch verteilte Grauwerte aufweist. (Neumann, 2004, S. 29)
In Abbildung 17 kann ergänzend dazu ein HSV-Bild mit dazugehörigen Histogrammen für jeden einzelnen Kanal betrachtet werden.

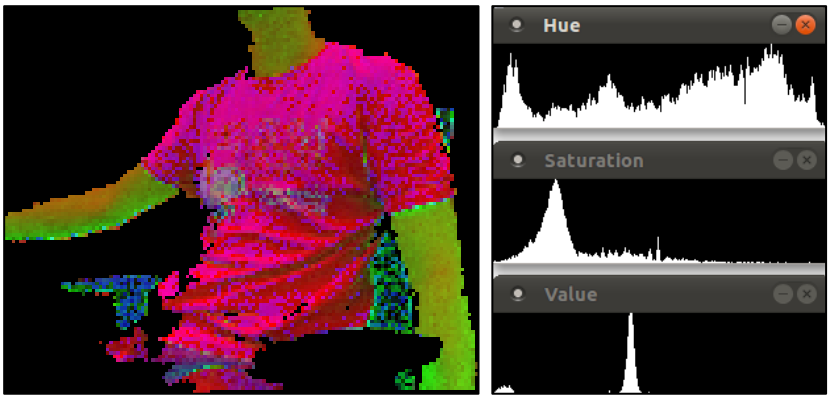

Abbildung 17: Farbhistogramm aus einem HSV-Bild

In der Version 2 des Gesamtsystems basiert die Detektion der Person vor allem auf den Farbenwerten seiner Kleidung. Hierfür ist die Verwendung des HSV-Histogramms essentiell (siehe). Aus diesem Grund wird die Histogrammfunktion bei der Erstellung des Initialisierungshistogramms 8.2.6 und beim Particle Filter 8.2.11 der Version 2 des Gesamtsystems eingesetzt.

3.2.9 Histogrammvergleiche

Wie im letzten Kapitel bereits erläutert, können aus Bildern, Histogramme erstellt werden. Wird nun davon ausgegangen, dass bereits ein Initialisierungshistogramm vorhanden ist, muss nun noch eines für ein aktuelles Bild erstellt werden. Eine Überprüfung, ob nun beide Histogramme zusammenpassen, also ob die Vorlage mit dem aktuellen Bild übereinstimmt, kann durch die OpenCv Funktion CompareHist realisiert werden. Dazu werden die beiden Histogramme mithilfe verschiedener Berechnungsmethoden miteinander verglichen. Vor allem Unterschiede in der Rechengeschwindigkeit sind ausschlaggebend für die Auswahl der Methoden. Soll eine Vorlage in einem Bild gefunden werden, so kann von dieser Vorlage zunächst ein Initialisierungshistogramm erstellt werden. In einem Bild wird dann ein Bereich betrachtet, welcher die gleiche Größe besitzt wie die Vorlage. Von diesem aktuellen Ausschnitt wird erneut ein Histogramm erstellt. Die Parameter bleiben dabei dieselben wie bei der Erstellung des Initialisierungshistogramms. Zum Vergleich beider Histogramme stehen vier Methoden zur Verfügung, welche unterschiedliche Ergebnisse liefern (Tabelle 5):

Correlation	$d_{\text{correl}}(H_1, H_2) = \frac{\sum_i H_1'(i) \cdot H_2'(i)}{\sqrt{\sum_i H_1'^2(i) \cdot H_2'^2(i)}}$	Wertebereich: [0-1]
Chi-Square	$d_{\text{chi-square}}(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)}$	Wertebereich: [0-2]
Intersection	$d_{\text{intersection}}(H_1, H_2) = \sum_i \min(H_1(i), H_2(i))$	Wertebereich: [0-1]
Bhattacharyya	$d_{\text{Bhattacharyya}}(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i) \cdot H_2(i)}}{\sqrt{\sum_i H_1(i) \cdot \sum_i H_2(i)}}}$	Wertebereich: [0-2]

Tabelle 5: CompareHist-Methoden, ihre dazugehörigen Formeln und Wertebereiche (nach OpenCv, 2011)

Tests ergaben, dass die Methode Intersection, am besten geeignet ist zwei Histogramme miteinander zu vergleichen. Die Berechnungen sind, im Vergleich zu den anderen Methoden, relativ simpel, was zu einer geringen Rechenzeit führt. Des Weiteren sind sie stabiler und aussagekräftiger, da sie von 0-1, je nach Übereinstimmungsgrad, jeden Wertebereich einnehmen können. Wie in der Formel in Tabelle 5 zu sehen ist, wird immer der minimalste Wert aus den beiden verglichenen Punkten in den Histogrammen ausgewählt. Dabei werden alle Werte des ersten Histogramms mit denen des zweiten Histogramms verglichen. Aus der Summe aller Ergebnisse errechnet sich schließlich die

Wahrscheinlichkeit, dass beide Histogramme zusammenpassen. Der Histogrammvergleich in Abbildung 18 ergab z.B. eine Übereinstimmung mit der Intersection Methode von 0,88.

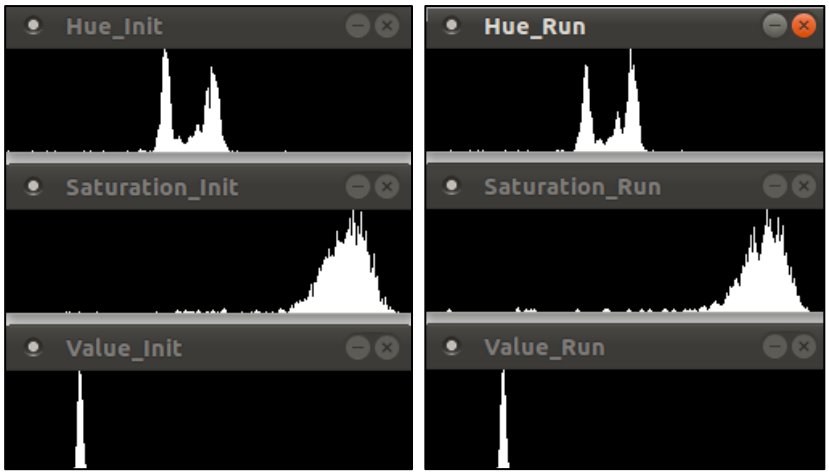

Abbildung 18: Histogrammvergleich mit der CompareHist-Funktion
links, Initialisierungshistogramm; rechts, aktuelles Histogramm

Vor allem in der Version 2 des Gesamtsystems findet diese Funktion ihre Verwendung. Neben ihrem Einsatz in der Initialisierungsphase, nutzt der Particle Filter die Ergebnisse aus dem Histogrammvergleich, um die Person immer verfolgen zu können.

3.2.10 Color Histogram Matching

Werden nun die Funktionen zur Histogrammerstellung, Histogrammvergleich und Template matcher kombiniert, ergibt sich ein Color Histogram Matching. Dieser vergleicht im speziellen die Farben der Vorlage mit denen des aktuellen Bildes. Dabei wird aus dem Vorlagenbild ein RGB-Histogramm (Kapitel 3.2.8) gebildet und dieses anschließend im Suchbild gesucht. Der Vorteil zum reinen Template matching liegt darin, dass dieses Verfahren invariant gegen Rotation oder Verkleinerung ist und somit die Vorlage stabiler im Bild gefunden werden kann. Wie bereits beim Template matcher erwähnt (Kapitel 3.2.6), besteht hierbei erneut das Problem, dass die Vorlage durch Verschiebung eines Suchfensters über das ganze Bild gesucht wird. Dies kann je nach Bildgröße durchaus zeit- und rechenintensiv sein.

Abbildung 19: Color Histogram Matching auf ein aktuelles Bild angewandt,
weißer Kasten: gefundenes rotes Objekt

In Abbildung 19 ist das Ergebnis des durchgeführten Beispiels zu sehen. Dabei wurde versucht, ein rotes Objekt im Bild zu finden und anschließend auch seine Bewegung zu verfolgen. Dies funktionierte solange gut, bis das Objekt zu lange verdeckt und somit ein anderes rotes Objekt im Bild gefunden wurde. Dieses Verfahren betrachtet somit alle roten Gegenstände im Bild als potentielle Ziele. Hat ein Objekt ein ähnliches Farbspektrum wie die Vorlage, ist es nicht mehr eindeutig unterscheidbar, welches jetzt das richtige ist.
Diese Funktion kommt in abgeänderter Form in der Version 2 des Gesamtsystems zum Einsatz. Dabei wird nicht, wie in diesem Beispiel gezeigt, eine einzige Farbe gesucht, sondern ein ganzes Farbspektrum. Dadurch ist es möglich eine Person in einem Bild stabiler zu verfolgen.

3.2.11 Flood Fill

Zum Zeitpunkt der Initialisierung ist es wichtig eine Zielperson komplett zu erfassen und sie somit für bestimmte Funktionen wie Template matcher oder Histogrammerstellung zur Verfügung zu stellen. Bei der FloodFill-Funktion handelt es sich um einen Algorithmus zur Erfassung aller zusammenhängenden Pixel in einem bestimmten Bereich. Somit können z.B. alle Pixel erfasst werden, welche annähernd die gleiche Farbe besitzen, oder sich, wie in diesem Fall, ausreichend intensiv von einer anderen Farbe unterscheiden.

Abbildung 20: Aufnahmen mit der FloodFill-Funktion
links, das gefilterte RGB-Bild
rechts, die erfasste Vase

Wie im Beispielbild (Abbildung 20) zu sehen ist, wurde eine Vase mit der FloodFill-Funktion erfasst. Der Hintergrund ist in diesem Falle komplett Schwarz, was zu einer sehr guten Abgrenzung zu den Farben der Objekte führt. Hierbei wurden die Parameter der Funktion so eingestellt, dass sie alle zusammenhängenden Punkte als ein ganzes Objekt erfassen. Wie in Abbildung 20-links zu sehen ist, wird ein Rechteck um dieses Objekt gelegt, welches automatisch seine Breite und seine Höhe definiert.
Die FloodFill-Funktion ist somit ein überaus praktisches Tool, um eine Person eindeutig erfassen zu können. Aus diesem Grund wird sie auch in der Version 2 des Gesamtsystems eingesetzt. Sollten sich in der Initialisierungsphase weitere Objekte in der unmittelbaren Nähe der Person befinden, werden diese nicht in die Erstellung des Initialisierungshistogramms miteinbezogen.

3.2.12 Scale Invariant Feature Transform

Des Weiteren gibt es die Möglichkeit durch die Detektion und den Vergleich von besonders markanten Punkten (Interessenspunkte) eine Person in einem Bild wiederzufinden. Eine Funktion, welche diese Interessenspunkte miteinander vergleicht, ist die SIFT-Funktion (SIFT steht für Scale Invariant Feature Transform). Dies geschieht unter anderem unabhängig von ihrem Skalierungsfaktor, dem Bildrauschen und der Beleuchtungssituation.
Zunächst startet die Trainingsphase mit der Erstellung einer Vorlage. Da es bei jedem Bild, je nach Qualität des Objektives, zu einem gewissen Rauschen kommt, muss dieses im Vorfeld weitestgehend beseitigt werden. Dies wird durch eine Filterfunktion namens Gaussian Blur (dt.: Gaußsche Unschärfe) erzielt. Der Filter reduziert dabei den Schärfegrad des Bildes und macht es dementsprechend unscharf. Was für das menschliche Auge möglicherweise nicht mehr zu erkennen ist, verhilft dem bereits erwähnten Kantenerkennungsoperator Difference of Gaussian (Kapitel 3.2.1) besonders markante Kanten zu identifizieren. An ihnen lassen sich die besten Interessenspunkte erkennen, da hierbei der Kontrastwert am höchsten ist. Diese Interessenspunkte werden

sowohl bei der Vorlage als auch bei dem zu überprüfenden Bild gesucht. Korrespondierende Punkte zwischen den beiden Bildern, können mithilfe einfacher graphischer Linien visualisiert werden.

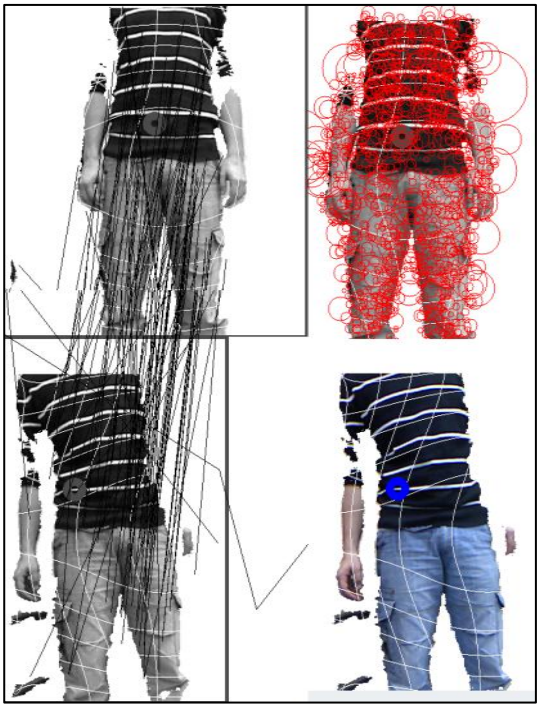

Abbildung 21: Der SIFT-Algorithmus in der Praxis
oben-links: Initialisierungsbild, oben-rechts: Initialisierungsbild mit Interessenspunkten
unten-links: gefundene Übereinstimmungen im aktuellen Bild, unten-rechts: aktuelles Bild

Auch wenn die SIFT-Funktion durchaus in statischen Bildern sehr gute Ergebnisse liefert, kann das System in einer Echtzeitumgebung nicht eingesetzt werden. Abbildung 21 zeigt dabei ein durchgeführtes Beispiel, bei dem eine Person verfolgt werden sollte. Die Berechnungen der Interessenspunkte und der Versuch die Vorlage im aktuellen Bild zu finden sind dabei so rechenintensiv, dass sie das gesamte System von idealerweise 30 Bildern/s auf 2 Bildern/s verlangsamt. Auch kommt es zu Problemen in der Detektion und Verfolgung der Person, wenn diese zu wenig texturierte Kleidung trägt, was zu entsprechend wenig Interessenspunkten führen würde. Die meisten SIFT-Objekterkennungsverfahren gehen des Weiteren von nahezu starren Objekten aus, weswegen der Einsatz zur Personendetektion nochmals erschwert wird. Aus diesem Grund findet die SIFT-Funktion keine Verwendung im Gesamtsystem.

3.3 Zusammenfassung der Ergebnisse

In den letzten Kapiteln wurde ein Überblick über alle getesteten Algorithmen der OpenCv-Bibliothek geliefert. Nicht alle hier dargestellten Funktionen finden auch Einzug in das endgültige Programm. Vor allem die Funktionen welche auf einer automatischen

Kantenerkennung basieren, versagen bei einer Personendetektion und -verfolgung, da die Ergebnisse nicht aussagekräftig genug für eine eindeutige Unterscheidung zwischen einer Person und einem Stuhl sind. Besonders die Funktionen rund um die Histogrammerstellung und den Histogrammvergleich scheinen vielversprechend für die weitere Verwendung im Gesamtsystem zu sein. Tabelle 6 fasst nochmal alle betrachteten Methoden zusammen:

Methode/ Einsatz	**Pro**	**Contra**
Kantenerkennung/ nein	- schnell - Kanten werden robust erkannt - viele Einstellmöglichkeiten	- keinerlei Aussage über das erfasste Objekt (Mensch, Stuhl)
Bewegungserkennung/ nein	- Schnell implementiert - Nicht sehr rechenintensiv - Für einfache Anwendungen durchaus geeignet	- Kamera muss statisch sein - Personen müssen sich bewegen - (noch) keine Unterscheidung zwischen einer Person und einem Auto
Ellipsen finden/ nein	- Ellipsenförmige oder runde Objekte können relativ gut gefunden werden	- Für die Detektion von Beinen nicht geeignet
Konturen finden/ nein	- Die markantesten Kanten werden erfolgreich gefunden	- keinerlei Aussage über das erfasste Objekt (Mensch, Stuhl)
Region-von-Interesse/ ja > Version 1 > Version 2	- Schnell implementierbar - Reduziert die zu betrachtende Bildinformationen, sodass eine Funktion (Particle Filter) deutlich beschleunigt wird	- Bereich muss vorher definiert werden - Eventuell können wichtige Informationen herausgeschnitten werden
Farbhistogramme/ ja	- schnell implementierbar - geben schnell Aussage über	- können teilweise ungenaue Ergebnisse

> Version 2	alle vorkommenden Farbwerte in einem Bild	liefern
Histogrammvergleich/ ja > Version 2	- zum Vergleich zweier Histogramme sehr gut einsetzbar - schnell implementierbar	- Je nach Anwendungsfall und verwendeter Methode kann es zu Fehlern in der Berechnung kommen - Je nach Methode zeitintensiver
Template matcher/ nein	- Relativ gute Erkennung einer Vorlage in einem Bild	- Teilweise sehr zeit- und rechenintensiv bei einem Echtzeitsystem
FastTemplateMatcher/ ja, in abgeänderter Form > Version 1	- Um ein vielfaches schneller als der reine Template matcher - Sehr viele Einstellungsmöglichkeiten	- Wird überaus langsam, wenn ein Objekt nicht mehr gefunden wird
Color Histogramm Matcher/ ja, in abgeänderter Form > Version 2	- Kann eine Farbe sehr gut in einem Bild wiederfinden	- Je nach Implementierung, durchaus zeit- und rechenintensiv
Flood Fill/ ja > Version 2	- Erfassung aller zusammenhängender Bildinformationen - Je nach Einstellung, Erfassung einer bestimmten Farbe	- Objekte welche zu nahe beieinander stehen, werden mit einbezogen - keinerlei Unterscheidung zwischen richtigem und falschem Objekt
Scale Invariant Feature Transform/ nein	- erkennt Objekte in statischen Bildern überaus gut	- zu langsam - zu rechenintensiv

Tabelle 6: Zusammenfassung aller betrachteten Methoden der 2-Dimensionalen Bildbearbeitung

4 3-Dimensionale Bildbearbeitung

Im Kapitel 3.2 wurden die verschiedenen Methoden zur Erfassung und Detektion einer Person in einem 2D-Bild erläutert. Um die genaue Position und den Abstand einer Person zum Roboter ermitteln zu können, müssen die Tiefeninformationen der Umgebung vorhanden sein. Diese Informationen sind in der 3D-Punktwolke und im Tiefenbild der Kinect Kamera gespeichert und können ohne weiteres ausgelesen werden. Dabei wurde bei der Erstellung der Version 1 des Gesamtsystems, der besondere Fokus auf die Verarbeitung der 3D-Punktwolke der Kinect Kamera gelegt. Um diese bearbeiten zu können wird das Robot Operating System (ROS) und die Point Cloud Library(PCL) verwendet. Darüber hinaus wurde für die Version 2 des Gesamtsystems das Tiefenbild der Kinect Kamera benutzt um die Distanz zur Zielperson ermitteln zu können. Im weiteren Verlauf dieses Kapitels, werden alle Funktionen und Algorithmen erläutert, welche dazu nötig sind. Ausgehend von den bereits erläuterten Grundlagen (4.1), werden im weiteren Verlauf die erweiterten Grundlagen, die betrachteten Methoden und die abschließende Zusammenfassung dieser vorgestellt.

4.1 Erweiterte Grundlagen

In diesen erweiterten Grundlagen, sollen vor allem die einzelnen benötigten Komponenten von ROS betrachtet werden. Außerdem erläutert ein Exkurs in Triangulationsbasierende Tiefenmesstechniken, den theoretischen Hintergrund zur Verwendung von Tiefenbildern.

ROS Packages

Ein ROS Package kann als komplettes Projekt angesehen werden, ähnlich einem Projekt bei Visual C++. In diesem werden alle Source-Codes und Header-Dateien abgespeichert, welche zwingend oder optional verfügbar sein müssen. Darüber hinaus werden in der Datei „manifest.xml“ alle Dependencies (Abhängigkeiten) für die korrekte Kompilierung und den ordnungsgemäßen Ablauf des Programms eingetragen. Diese Abhängigkeiten sind wiederum andere Packages, welche im Hauptverzeichnis der ROS-Installation entweder mit installiert, oder nachträglich hinzugefügt werden. Diese Packages beinhalten wiederum eigene Headerdateien und Codeelemente, welche verschiedene Funktionen erfüllen.

Das PCL-Package

Für dieses Projekt soll ein wichtiges Package hervorgehoben werden. Es handelt sich dabei um das PCL (Point Cloud Library)-Package und beinhaltet unter anderem, vielfältige Methoden zur Segmentierung, Filterung und zur Darstellung und Bearbeitung von 3D-Punktwolken. Mittlerweile gibt es dieses Package auch als Standalone-Installation, welches dieses nun von ROS unabhängig macht. Vor allem ist darauf zu achten die richtigen Header Dateien einzubinden und auch die richtige Codeschreibweise zu

verwenden. Mit entsprechenden Codezeilen werden nun beispielsweise Punktwolken in RGB- oder in einer vorher definierten Farbe dargestellt oder bearbeitet.

ROS Vorbereitung

Jedes ROS-Programm muss zu Beginn seiner Laufzeit einen eigenen Knoten erstellen und sich somit im Gesamtsystem registrieren. Von diesem Knoten können Informationen verschickt werden, oder er kann selber Informationen empfangen. Dieser erhält auch einen fix definierten Namen, welcher ihn einzigartig im System macht. Wird nun ein weiterer Knoten mit dem gleichen Namen erstellt, so wird der vorher erstellte sofort mit einer Fehlermeldung beendet. Durch diese Sicherheitsmaßnahme können sich niemals zwei gleiche Programme im System aufhalten und evtl. gleichzeitig versuchen, andere Variablen zu manipulieren.

Einen ROS Subscriber erstellen

Um überhaupt eine Punktwolke zu erhalten, muss ein Topic abonniert werden. Dieses wird vom Treiber der Kamera veröffentlicht und das Programm kann nun auf dieses zugreifen. Somit können jegliche Informationen, seien es nun Nachrichten, Zahlen, oder eben ganze Bildinformationen wie z.B. das RGB-, oder Tiefenbild innerhalb des Systems versendet und empfangen werden. Mit der entsprechenden Codezeile können nun die Informationen der RGB-Punktwolke empfangen und weiterverarbeitet werden.

OpenNI-Camera Library von Primesense

Kurz nach dem Verkaufsstart der Kinect für die Xbox360 wurde der Quellcode dieser von der Herstellerfirma Primesense veröffentlicht und für jeden zugänglich gemacht. Dadurch ist es nun möglich, die Kinect auch ohne eine Xbox360 zu betreiben und sie dadurch auch für andere Anwendungsfälle nutzbar zu machen. Vor allem im Bereich der Robotik findet diese preisgünstige Kamera-Alternative einen guten Anklang. Durch die Freigabe des Codes konnte das ROS-Team die Kinect schnell in das bestehende System integrieren und für alle ROS operierenden Plattformen zugänglich machen. Dazu muss lediglich das OpenNI-Package installiert und dieses dann im Programm eingepflegt werden. Die Installationsanleitung dazu kann hier gefunden werden.

Exkurs Tiefenmesstechniken durch Stereoskopie

Bei der Betrachtung eines Objektes aus unterschiedlichen Positionen, welche durch einen Basisvektor b voneinander getrennt sind, ergeben sich unterschiedliche Blickwinkel. Die sich daraus ergebende Verschiebung der Bildebene wird als Disparität oder Parallaxe bezeichnet und dient zur Bestimmung der Entfernung des Objektes. Die Tiefenmesstechniken beruhen dabei auf 2 Prinzipien.

Ein Aufbau zweier Bildsensoren, wie sie auch in der Natur vorkommen, wird Stereosystem genannt. Anders als bei üblichen Stereokamerasystemen, welche zwei optische Kameras (zwei Bildsensoren) verwenden, besitzt die Kinect Kamera einen Infrarot Projektor und eine Infrarotkamera (ein Projektor, ein Bildsensor). Das Prinzip worauf die Tiefenwertberechnung dieser Kamera basiert, ist dasselbe, weswegen Formeln und Erklärungen auch für diese gelten. Grundsätzlich wird der Abstandsvektor b zwischen zwei Kameras, welche auf parallelen optischen Achsen platziert werden, als stereoskopische Basis bezeichnet. Durch die versetzte Betrachtung erscheint der Gegenstand nun an unterschiedlichen Positionen in der Bildebene.

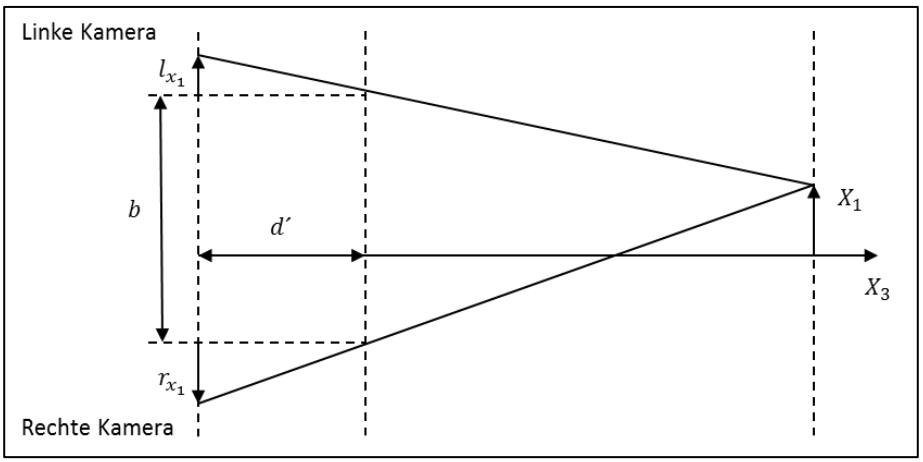

Abbildung 22: Schematische Darstellung einer Stereokameraanordnung (nach Jähne, 2005, S.232)

Aus der Grafik in Abbildung 22 lässt sich nun folgende Formel entwickeln (Formel 1):

$$p = {x_1}^r - {x_1}^l = d' \frac{X_1 + \frac{b}{2}}{X_3} - d' \frac{X_1 - b/2}{X_3} = b * \frac{d'}{X_3}$$

Formel 1: Berechnung der Parallaxe p (Jähne, 2005, S.232)

Dabei ist die Parallaxe p umgekehrt proportional zur Entfernung X_3 des Objektes und des Weiteren direkt proportional zur stereoskopischen Basis und der Brennweite der in diesem Fall verwendeten Objektive. Ist $X_3 = 0$, so hat das Objekt eine unendliche Entfernung. Außerdem kann für weit entfernte Gegenstände der Distanzvektor $d' \approx f$ angenommen werden. Dies führt zu dem Schluss, dass eine Entfernungsabschätzung umso schwieriger wird, desto weiter ein Objekt entfernt ist. Die absolute Sensitivität für eine Tiefenbestimmung nimmt demnach mit dem Quadrat der Entfernung ab. Bei einem Beispielaufbau eines Stereosystems mit einer stereoskopischen Basis von 200 mm, einer Brennweite des Objektivs von 100 mm und einer Entfernung von 10 m zum Objekt, beträgt die Veränderung der Parallaxe etwa 200 μm/m (dies entspricht ca. 20 Pixel/m). Bei einer Entfernung von 100 m zum Objekt beträgt sich nur noch 2 μm/m (ca. 0,2 Pixel/m).

Bei der Parallaxe handelt es sich um eine Vektorgröße, welche sich immer parallel zur stereoskopischen Basis b befindet. Dies hat den Vorteil, dass die Richtung der Parallaxe relativ leicht bekannt ist, allerdings auch den Nachteil, dass deren Betrag nicht so leicht bestimmt werden kann. Bei einem Bildbereich ohne Struktur können keine Verschiebungen bestimmt werden, weil keine Verschiebungen in den Grauwerten in diese Richtung erfolgen. (Jähne, 2005, S.231ff)

Exkurs Tiefenmesstechniken durch die Signallaufzeit

Die Berechnung der Tiefe aus der Laufzeit eines Signals, wird vor allem bei der Verwendung des Laser Range Sensors, zur Navigation und Kollisionsvermeidung benötigt. Dabei legt eine Strahlung, welche von der Position des Sensors ausgesendet wird, ihre Strecke um die Entfernung zum Objekt zurücklegen zu können zweimal zurück. Diese zurückgelegte Strecke ergibt eine Verzögerungszeit vom Aussenden bis zum erneuten Erhalt des Signals. Daraus ergibt sich folgende Formel (Formel 2):

$$\tau = \frac{2z}{c}$$

Formel 2: Berechnung der Tiefe aus der Verzögerungszeit
(Jähne, 2005, S.239)

Dabei steht c für die Ausbreitungsgeschwindigkeit der Strahlung. Der statistische Fehler der Tiefenmessung ist hierbei unabhängig von der Distanz zum Objekt. Dieser hängt nur von der Messgenauigkeit der Verzögerungszeit ab und stellt somit einen großen Vorteil im Vergleich zu Triangulationsverfahren dar (Formel 3).

$$z = \frac{c}{2} * \tau \Rightarrow \sigma_z = \frac{c}{2} * \sigma_\tau$$

Formel 3: Berechnung der Tiefe aus der Verzögerungszeit in Abhängigkeit der Messgenauigkeit
(Jähne, 2005, S.239)

Bei einer Berechnung der Laufzeit wird in erster Linie auf die Pulsmodulation, also die Messung der Verzögerungszeit zwischen dem Senden und dem Empfangen eines kurzen Signalpulses, zurückgegriffen. Bei elektromagnetischen Wellen hingegen ist die Vermessung der Verzögerungszeit noch eine Herausforderung, da die Lichtgeschwindigkeit c (3*10^8 m/s) lediglich eine Verzögerungszeit von 6,7ns pro Meter aufweist.

Neben der Pulsmodulation kann auch die Laufzeit durch periodische Modulation der Signalamplitude errechnet werden. Hierbei wird die Laufzeit durch die Phasenverschiebung zwischen dem aus- und eingehenden Signal nach folgender Formel gemessen (Formel 4):

$$z = \frac{c}{4\pi v}\varphi \Rightarrow \sigma_z = \frac{c}{4\pi v} * \sigma_\varphi$$

Formel 4: Berechnung der Laufzeit aus der Phasenverschiebung (Jähne, 2005, S.240)

Die Variable v steht dabei für die Frequenz der Modulation. Die Tiefe kann nun berechnet werden, da die Phase eindeutig nur in im Bereich von $\pm\pi$ gemessen werden kann (Formel 5).

$$\Delta z = \frac{c}{2v} = \frac{cT}{2}$$

Formel 5: Berechnung der Tiefe aus der Phasenverschiebung (Jähne, 2005, S.240)

Die Berechnung der Tiefe aus einer periodischen Modulation hat daher den größten Nachteil im begrenzten Tiefenbereich. Durch Zufallsmodulation der Signalamplitude, kann dieses Problem gelöst werden. Dadurch werden die Vorteile der hohen Auflösung der periodischen Modulation mit dem großen Tiefenbereich der Pulsmodulation kombiniert. (Jähne, 2005, S.239ff)

Image Transport

Durch die Integration von ROS können nun die eingebauten Pakete verwendet werden um den Roboter zu steuern. Die Kommunikation in ROS funktioniert über Nodes (Knoten), welche von Programmen erstellt und wiederum von anderen Programmen abonnieren werden können. Dieser Aufbau ist in Abbildung 23 mithilfe von den verschiedenen Knoten dargestellt.

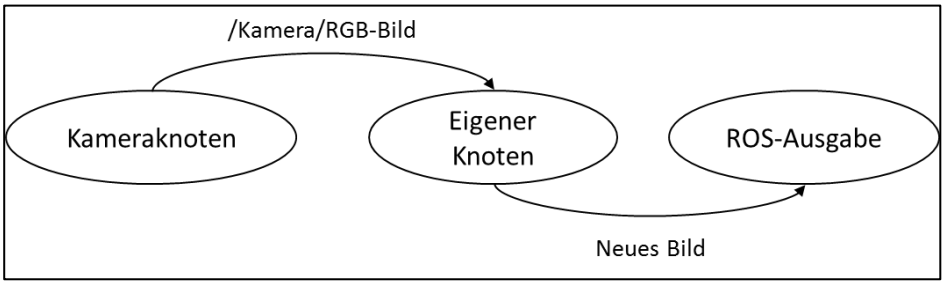

Abbildung 23: Beispiel eines ROS-Programms in Knotenansicht

Wurden die Daten des ersten Knotens veröffentlicht, kann nun dieser Knoten abonniert und die Information ausgewertet werden. Dabei wird das empfangene Bild im eigenen Knoten bearbeitet und schließlich in einem neuen Fenster über die ROS-Ausgabe angezeigt (Abbildung 24). Die gleiche Methode kann verwendet werden um die Live-Bilder der Kamera oder schlussendlich auch die Informationen der 3D-Punktwolke auszugeben.

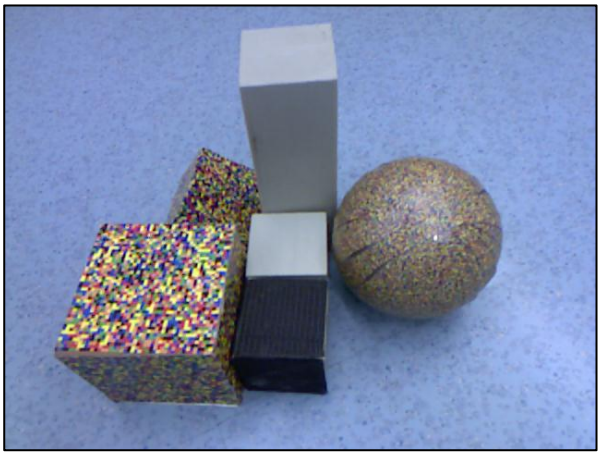

Abbildung 24: Ausgabe eines Bildes über einen eigenen Knoten

4.2 Betrachtete Methoden

Wie bereits im 2D-Bildbearbeitungskapitel (3.2) erklärt, werden auch in diesem Kapitel die verschiedenen betrachteten Methoden vorgestellt und ihre Funktion im Detail erläutert.

4.2.1 Die 3-Dimensionale Punktwolke

Neben der Detektion der Zielperson durch die 2D-Bildverfahren (3.2), ist die Ermittlung des Abstandes der Person zur Kamera ist essentiell für die korrekte Steuerung des Roboters. Dazu müssen die Tiefeninformationen der Umgebung vorhanden sein, welche in der 3D-Punktwolke, bereitgestellt durch die Kinect Kamera, gespeichert sind. Wie bereits im Kapitel (2.2.3) des Kinect-Aufbaus erläutert, wird aus der Kombination von Infrarot-Projektor und Infrarot-Kamera ein Tiefenbild erstellt, welches den genauen Abstand jedes Punktes zur Kamera im sichtbaren Bereich aufzeigt. Daraus können nun die benötigten Informationen herausgelesen werden, in welcher Entfernung sich die Zielperson vom Roboter befindet. Abbildung 25 zeigt dabei zwei Beispielaufnahmen der 3D-Punktwolke, wobei links eine Person und rechts ein Beine erfasst wurde.

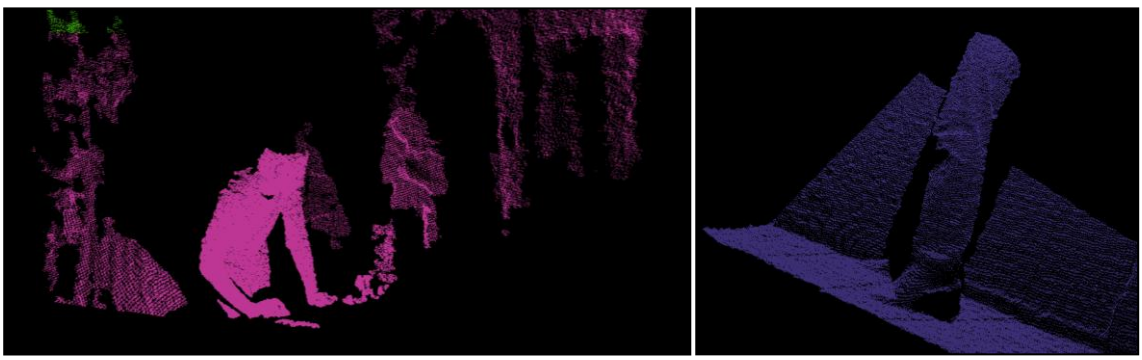

Abbildung 25: Beispiele der 3D-Aufnahme

Die 3D-Punktwolke ist die grundlegende Komponente in der Version 1 des Gesamtsystems. Auf ihr basieren alle weiteren Funktionen und verwendeten Algorithmen.

4.2.2 Filterungsmethoden der 3-Dimensionalen Punktwolke

Neben der Zielperson, werden sich in einem vollständigen Bild, auch unerwünschte Informationen befinden. Wird z.B. in der Initialisierungsphase definiert, dass die Person sich in einem Abstand von maximal 1.5 m zur Kamera befinden darf, können alle Bildinformationen, welche einen höheren Abstand haben, gelöscht werden. Durch die Verwendung der 3D-Punktwolke und der Point Cloud Library (PCL) stehen zu diesem Zweck verschiedene Filterfunktionen zur Verfügung, welche nicht benötigte Informationen herauslöschen können. Im weiteren Verlauf sollen diese Filter und ihre speziellen Funktionen betrachtet und die für die Version 1 des Gesamtsystems geeignetsten ermittelt werden.

Passthrough-Filter

Sollen nicht benötigte Informationen ab einem bestimmten Abstand herausgefiltert werden, bietet das PCL-Package im ROS-System für diesen Zweck den Passtrough-Filter an. Jeder sichtbare 3D-Punkt beinhaltet die XYZ-Positionsdaten, wodurch er klar im Raum definiert wird. Somit kann jeder Punkt, welcher sich außerhalb des definierten Bereichs liegt, gelöscht werden. Dies würde einem 3D-Region-of-Interest entsprechen. Diese Filterung kann sowohl für einen Abstand in Z-Richtung von z.B. 1.5 m oder für eine seitliche Filterung ausgehend vom Bildmittelpunkt um ±0.5 m durchgeführt werden.

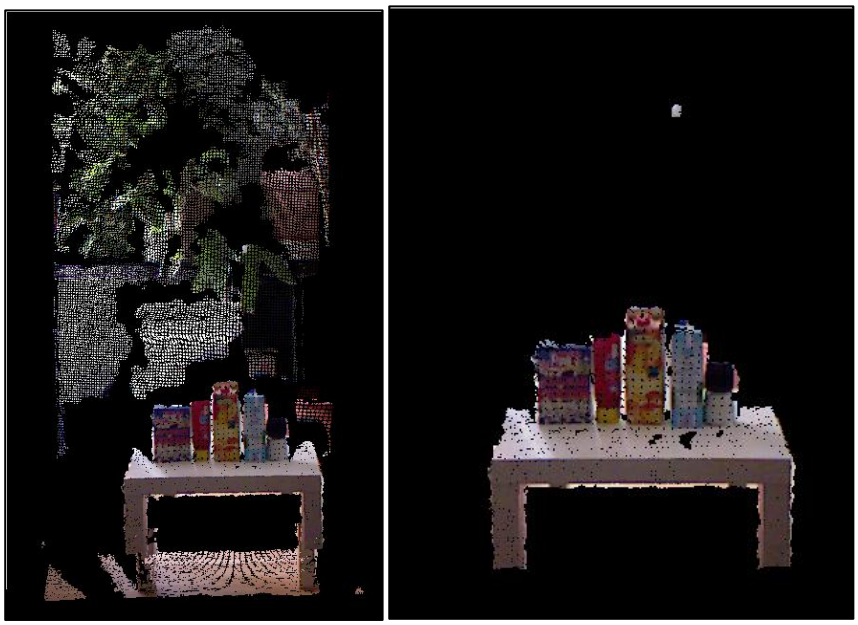

Abbildung 26: Ansichten der Seiten- (links) und Abstandsfilterung (rechts)

Wie in Abbildung 26 zu sehen ist, beinhaltet die 3D-Punktwolke nach der Filterung nur noch die benötigten Informationen. Durch die Filterung der unnötigen Informationen verringert sich zudem die Größe der Punktwolke. Dies hat den Nebeneffekt, dass Berechnungen von nachfolgenden Funktionen beschleunigt werden. Aus diesem Grund wird der Passthrough-Filter in der Version 1 des Gesamtsystems eingesetzt.

Planare Segmentation

Eine weitere Reduktion der Informationsmenge der Punktwolke, welche optional angewendet werden kann, ist die Filterung von Flächen. Dabei werden alle relativ zusammenhängenden Punkte in einer Ebene als Fläche angesehen und können nun verändert werden. Die PCL-Funktion beinhaltet dabei verschiedene vordefinierte Modelle zur Filterung, wie z.B. zylindrische oder runde Objekte. Die für diese Zwecke wichtige Funktion, ist die Filterung der planaren Flächen, wie es Tische oder Fußböden haben. Da sich nicht alle Punkte des Bodens wirklich in einer perfekten Ebene befinden, kann noch zusätzlich eine Toleranz eingesetzt werden. Die Ergebnisse der planaren Filterung können in Abbildung 27 betrachtet werden.

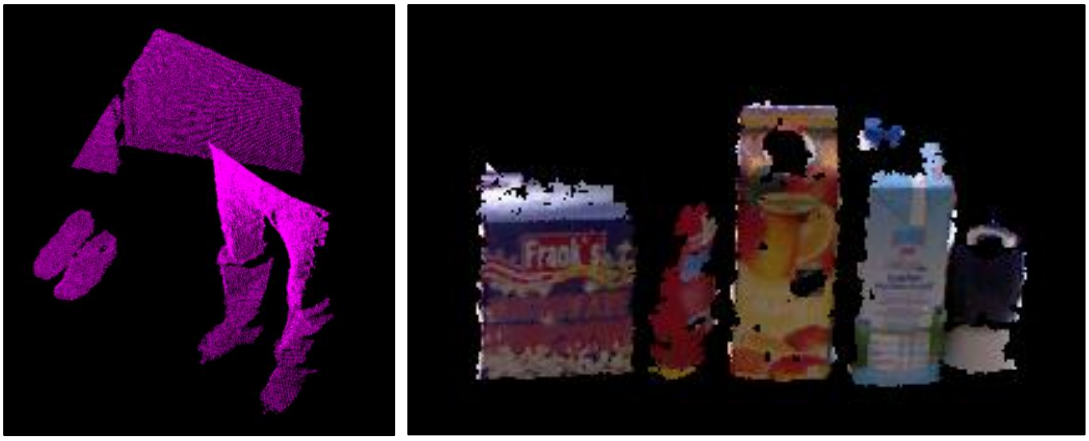

Abbildung 27: Ansichten von der Filterung des Bodens (links) und des Tisches (rechts)

Filterung zylindrischer Flächen

Neben planaren Flächen, können auch zylindrische Flächen herausgefiltert werden. Diese haben eine bestimmte Form und können so leichter im Bild gefunden werden. Der Filter kann nun die zylindrische Form aus dem Bild herausschneiden und als eigenes Objekt segmentieren. Der Hintergrund fällt dabei weg. Eine solche Filterung wäre überaus praktisch um die Beine einer Person aus der gesamten Szenerie herauszufiltern. Probleme tauchen dann auf, wenn die Person sich bewegt, einen Mantel oder einen Rock zum Zeitpunkt der Erfassung hatte oder ein anderes Objekt vor der Person steht, da dadurch eine eindeutige Identifizierung nicht mehr gewährleistet werden kann.

Voxel Filter

Eine Punktwolke besteht, je nach erfasster Szenerie, aus zehntausenden von Punkten. Diese Informationen darzustellen und zu verarbeiten ist überaus rechenintensiv. Deswegen enthält das PCL-Package einen Filter, welcher die Menge an Punkten reduzieren kann. Dabei wird die Punktwolke nur ausgedünnt. Nach der Filterung fallen somit nicht große Bereiche des Bildes weg, sondern nur bestimmte, redundante Punkte, welche für die Gesamtdarstellung nicht von Bedeutung sind. Diese Filterung erscheint auf den ersten Blick überaus praktisch, ist aber um einiges rechenintensiver als die Punktwolke komplett zu lassen. Jede Filterung bedarf einiger Rechenzeit und bei dieser ist diese zu groß, als dass sie sinnvoll in das Programm integriert werden konnte.

4.2.3 Tiefenbild Filterung

Eine andere Möglichkeit, eine Person aus dem Bild heraus zu filtern als die bereits im Kapitel 4.2.2 erwähnte Punktwolkenfilterung, besteht in der direkten Verwendung des Tiefenbildes. Dies wird automatisch vom Treiber der Kamera erstellt und steht im entsprechenden Knoten zur Verfügung. Wie bereits im Kapitel 4.1 erläutert, kann die gleiche Methodik des Image Transports genutzt werden um Bilder der Kamera zu erhalten. Um in diesem Fall die Person aus dem Bild heraus segmentieren zu können, werden alle Bildpunkte im Tiefenbild mit einer Entfernung größer als 1,5 m gesucht, anschließend der Wert auf 0 gesetzt und im Tiefen-Bild schwarz eingefärbt.
Dadurch kann eine saubere Segmentierung erzielt werden. Im Vergleich zu der Verwendung der 3D-Punktwolke, ist diese Methodik weitaus schneller und weniger rechenintensiv.

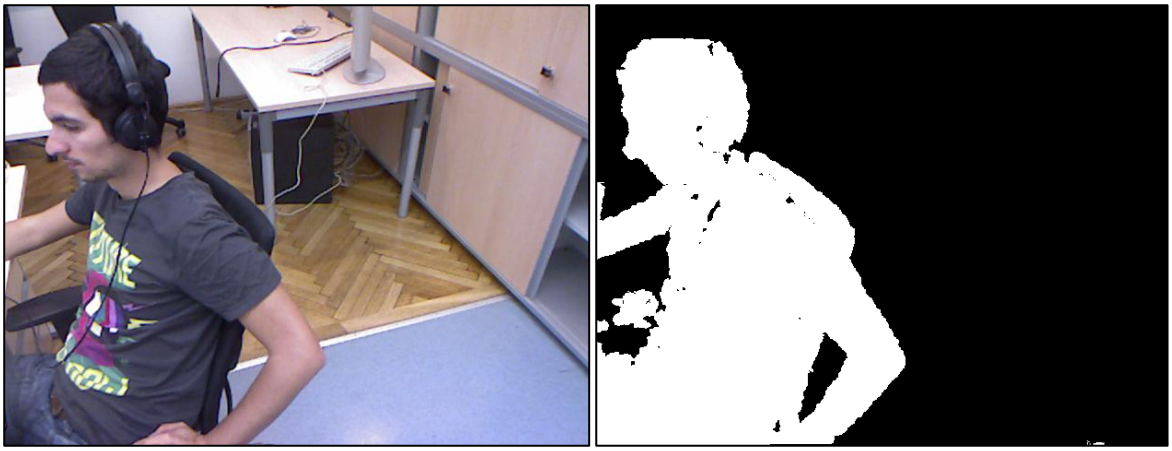

Abbildung 28: Gefiltertes Tiefenbild

Diese Methode lässt sich nun auch äquivalent auf das Farbbild anwenden. Dazu werden die Farbinformationen aller Punkte, welche sich noch innerhalb der Toleranz befinden, in ein neues Farbbild projiziert und ergeben ein segmentiertes Farbbild (Abbildung 28).

Abbildung 29: Gefiltertes Farbbild

Des Weiteren besteht die Möglichkeit die Tiefe eines Bildes farblich darzustellen. Dadurch werden Bereiche. Welche sich z.B. zu nahe an der Kamera befinden, Rot dargestellt und Bereiche, welche weiter weg sind, erhalten einen bläulicheren Farbton (Abbildung 29). Außerdem können in Abbildung 30 die farblich dargestellten Tiefeninformationen des Bildes in Abhängigkeit ihrer Distanz betrachtet werden.

Abbildung 30: Tiefeninformationen des Bildes farblich nach Abstand dargestellt

4.3 Zusammenfassung der Ergebnisse

Die verschiedenen durchgeführten Experimente haben, wie bereits im Kapitel der 2D-Bildverarbeitung gezeigt, dass nur die Kombination aus diesen Funktionen zu einem funktionierenden System beitragen. Jede Funktion ist für sich gesehen sehr gut, doch kann sie in einer komplexen Umgebung schnell versagen. Während der Experimentierphase, zeichnete sich bereits ab, dass die Verwendung des Tiefenbildes, in punkto Rechengeschwindigkeit und Programmierarbeit der 3D-Punktwolke weitaus überlegen ist. Nichtdestotrotz war die Implementierung der 3D-Punktwolke in der Version 1 des Gesamtsystems schon so weit vorangeschritten, dass dieses Programm als eigenes

System angesehen werden kann. Im Kapitel 8 werden schließlich beide Versionen im Detail vorgestellt und die einzelnen verwendeten Funktionen ausführlich beschrieben. Tabelle 7 zeigt dabei die betrachteten Methoden im Überblick:

Methode/ Einsatz	**Pro**	**Contra**
3D-Punktwolke/ ja > Version 1	- die einzelnen Punkte beinhalten zusätzlich die Orientierung - kann auf vordefinierte Filterfunktionen zurückgreifen	- Je nach Größe der Punktwolke, sehr zeit-und rechenintensiv
Passtrough-Filter/ ja > Version 1	- verringert die Anzahl der Punkte - dadurch wird kann das System beschleunigt werden	- Filterung kostet extra Zeit und Rechenleistung
Planare Segmentation/ nein	- schneidet nicht benötigte Böden oder Tische relativ einfach heraus - dadurch werden die Objekte (Beine, Tassen) freigestellt	- durchaus zeit-und rechenintensiv
Filterung zylindrischer / nein	- freigestellte Objekte werden ohne viel Programmieraufwand erkannt	- unbrauchbar bei der Detektion der Beine einer Person
Voxel Filter/ Nein	- verringert die Anzahl der Punkte in einer 3D-Punktwolke - in statischen Punktwolken durchaus ein Geschwindigkeitszuwachs	- in Echtzeitsystem überaus zeit-und rechenintensiv
Tiefenbild/ ja > Version 2	- schnell - einfach zu implementieren - weitaus weniger zeit-und rechenintensiv als die 3D-Punktwolke	- gibt keinerlei Informationen über die Orientierung eines Punktes - Besitzt keine vordefinierten Filterfunktionen

Tabelle 7: Zusammenfassung aller betrachteten Methoden der 3-Dimensionalen Bildbearbeitung

5 Particle Filter

Um Objekte in einem Kamerabild erkennen und verfolgen zu können wurden bereits einige Funktionen und Algorithmen vorgestellt (siehe Kapitel 3.2). Durch die Kombination dieser Funktionen kann die Stabilität bei der Erkennung einer Vorlage in einem Bild gesteigert werden. Der Particle Filter, nutzt dabei die Ergebnisse aus den verwendeten Funktionen, um, ähnlich dem Kalman Filter, eine Schätzung von sich veränderten Symbolzuständen (z.B. Position eines Objektes) durchzuführen.
Seinen Namen erhält der Particle Filter dadurch, dass immer ein Particle für jeweils eine Berechnung steht und die Position dieser meistens graphisch durch einen Punkt dargestellt wird. Dieser Punkt kennzeichnet meistens das Zentrum der Betrachtung in einem ROI. Dazu benötigt dieser vorher definierte Merkmale, welche er für seine Erkennung benutzen kann. Mit diesen und anderen Parametern, durchläuft die Funktion verschiedene Schritte um zu einem Ergebnis zu kommen.

Abbildung 31: Ansichten der Particles auf einem erfassten Bild

5.1 Aufbau des Particle Filters

Die Schritte zur Realisierung und Implementierung eines Particle Filters werden im nachfolgenden kurz aufgeschlüsselt und im weiteren Verlauf, im Detail erläutert:

1. initParticleFilter: Initialisierung des Particles Filters
2. resample: neue Particles werden anhand der berechneten Gewichtung erstellt
3. system: Zufallspositionierung der Particles, mit einer bestimmten Streuung an einem bestimmten Punkt
4. updateWeights: Gewichtungsberechnung für jedes Particle
    a. observationError: Fehlerberechnung für jedes Particle
5. normaliseWeights: die Summe aller Particles wird auf 1 gesetzt

initParticleFilter

Um einen Particle Filter optimal nutzen zu können, muss dieser entsprechend initialisiert werden. Dazu benötigt er ein Grundmodell, an welchem er sich orientieren kann und

verschiedene Parameter, welche ihm bei der späteren Ausführung helfen. Diese Parameter können Größe, Farbe oder Form eines Objektes sein. Dieses Modell wird dem Particle Filter zusammen mit der Anzahl der zu erstellenden Particles übergeben. Je nach Anwendungsfall kann eine Initialisierungsposition im Bild definiert oder alle Particles, zufällig über dem Bild verteilt werden.

resample

Beim resampling werden aus den n-aktuellen-Particles die n-Particles mit der höchsten Gewichtung ausgewählt (siehe updateWeights). Dabei wird ein Particle umso wahrscheinlicher gewählt, je größer sein Gewicht ist. Konkret wird der Bereich [0,1] in n-Bereiche unterteilt, wobei jeder Teilbereich einem Particle und die Größe des Bereiches dem Gewicht des Particles entspricht. Nun werden n Zufallszahlen in den Bereich gelegt. Landet eine Zufallszahl in einen Teilbereich so wird das entsprechende Particle ausgewählt. Größere Bereiche werden öfters gewählt und somit Particles mit größerem Gewicht, auch öfters ausgewählt. Dadurch können alle guten Particles von den schlechten getrennt werden. Dies wird benötigt um der nachfolgenden Berechnungsfunktion, nur die besten und richtigen Particles zu übergeben. Würde dieser Schritt übersprungen werden, würden auch falsche Particles in die Berechnung einfließen und das Ergebnis verfälschen. Nur die besten Particles überleben den resampling Prozess.

system

Ein Particle Filter besitzt normalerweise ein Bewegungsmodell. Basierend darauf führt er eine Vorhersage der nächsten Position durch. Dazu ist es nötig, möglichst viele wahrscheinliche Positionen abzudecken. Aus diesem Grund wird eine Zufallsfunktion eingesetzt, welche dem aktuellen Punkt ein gewisses Rauschen mitgibt. Bei der uniform-randomize Methode, werden alle Werte zwischen 0 und n (wobei n definiert werden muss) in Betracht gezogen und entsprechend eines Zufallsprinzip gewählt. Bei der normalform-randomizer Funktion, werden die Zahlen Normalverteilt. Das bedeutet, sie werden nach einer Gaußschen Glockenkurve verteilt, was entsprechend harmonischer und natürlicher ist. Das eingesetzte Rauschen definiert dabei die Größe bis wohin die Zufallszahlen erstellt werden.

updateWeights

In dieser Funktion wird für jedes Particle ein neues sogenanntes Gewicht ermittelt. Dazu wird jedes Particle in die observationError-Funktion übergeben. Je kleiner der dabei errechnete observation Error, desto größer ist sein Gewicht.

observationError

Um zu ermitteln wie gut oder schlecht das Gewicht eines Particles ist, muss dieses anhand seiner Merkmale berechnet werden. Dazu werden die aktuellen Bildmerkmale des

Particles erfasst und mit den Modellmerkmalen (siehe initParticleFilter) verglichen werden. Die Differenz zwischen diesen beiden wird als Fehler angesehen und gibt Aufschluss darüber, inwieweit das aktuelle Particle mit den geforderten Kriterien übereinstimmt oder abweicht.

normaliseWeights
Bei der normaliseWeights-Funktion, werden alle Gewichte soweit umgerechnet, dass sie in Summe 1 ergeben. Dies wird im weiteren Verlauf des Particle Filters für die resample-Funktion benötigt um gute Particles von schlechten unterscheiden zu können.

5.2 Fazit Particle Filter

Der Particle Filter ist eine sehr gute Methode um Objekte in einem Kamerabild zu verfolgen. Wie die eingehenden Tests mit verschiedenen Objekten gezeigt haben, reagiert er ausreichen schnell auf eine Positionsveränderung des Objektes im Bild. Anders als der FastTemplateMatcher aus Version 1 des Gesamtsystems, ist der Particle Filter um ein vielfaches schneller und besser bei der Verfolgung einer Person oder eines Objektes im Bild. Je nachdem, welche OpenCv Funktionen ihm übergeben werden, kann er ein Zielobjekt überaus genau verfolgen. Der Particle Filter ist darüber hinaus noch weitaus ausbaufähiger als es in dieser Testphase implementiert wurde. Durch die Möglichkeit einer Vielzahl von verschiedenen Algorithmen als Überprüfungsfunktionen zu implementieren, kann er ausgezeichnet in die Version 2 des Gesamtsystems integriert werden.

6 Drahtlose Systeme

Wie in den letzten beiden Kapiteln bereits ausgeführt, können Personen mittels einer Kamera erkannt und verfolgt werden. Die Detektion einer Person kann allerdings auch falsche positive Werte liefern. Das bedeutet, dass eine Person als die Zielperson erkannt wird, obwohl sie nicht die Person ist, welche verfolgt werden sollte. Um dies zu verhindern und die Detektion dadurch stabiler und besser zu machen, werden im Nachfolgenden einige Systeme betrachtet, welche auf diversen Funksystemen basieren. Die Idee dabei ist, dass die Zielperson einen Chip erhält, welche sie eindeutig identifizierbar macht.

Distanzberechnung

Das Problem, welches sich hieraus ergibt, ist die nicht vorhandene Möglichkeit, die genaue Position der Person zu bestimmen. Nur in einem System, bei dem ein ganzes Netzwerk aus Sendern und Empfängern vorhanden ist, besteht die Möglichkeit, dies zu ermitteln. Nichtsdestotrotz, kann die ungefähre Position aus der Signalstärke ermittelt werden. Dennoch ist sie ausreichend, um dem System die ungefähre Distanz der Person mitzuteilen. Für das Programm ist dieser Wert dahingehend von Bedeutung, da es daraus herleiten kann, ob sich eine Person nun vom Roboter weg bewegt hat und ob diese Information auch mit den Ergebnissen aus den Distanzberechnungen der Kamera-Funktionen übereinstimmt.

Im weiteren Verlauf dieses Kapitels wird die theoretische Verwendung verschiedener funkbasierten Systemen erläutert. Obwohl keines dieser Systeme in den ersten Prototypen verbaut wurde, können die theoretischen Grundlagen für Weiterentwicklung in nachfolgende Projekte verwendet werden. Des Weiteren schlüsselt Tabelle 8 die Grundlegenden Eigenschaften der drei Systeme: ZigBee, Bluetooth und WLAN, auf.

Parameter	Beschreibung	ZigBee, IEEE 802.15.4	Bluetooth, IEEE 802.15.1	WLAN, IEEE 802.11b
Topologie	P2P Sternnetz Maschennetz	X X X	X X	X X
Netzteilnehmer		>65000	8	32
Sprachkanäle			3	
Geschwindigkeiten		<300kByte		
Reichweite [m]	Gebäude Im freien	14 175	5-10 100	25-40 >100
Max. Datenrate [kBit/s]	Weltweit Amerika Europa	250 (2,4Ghz) 40 (915 MHz) 20 (868 MHz)	10^3	$11 * 10^3$
Anwendung (-automatisierung)	Heim- Industrie- Gebäude- Unterhaltungs-elektronik	X X X	X	X
Anbieter	Hardware Software Systemintegration	wenige wenige viele	viele viele viele	viele viele viele
Batterielebensdauer [d]		100-1000+	1-7	0,5-5
Kosten [Euro]	pro Modul	<3	≈3	2 * 3
Latenz		µs	µs	ms
Zielmarkt und Anwendung		Steuern, Überwachen, Automatisieren; Sensornetzwerk	Kabelersatz	Internetanwendung, Video
Anforderung an Systemressourcen		niedrig	mittel	hoch

Tabelle 8: Vergleich der Haupt-Funknetzwerke
(nach Gessler, Krause, 2009, S.249ff)

6.1 Identifikation mittels Radio Frequency Identification

Die RFID-Technology (Radio Frequency Identification) hat sich in den letzten Jahren immer mehr verbreitet. Durch den Einsatz der überaus günstigen und wiederverwendbaren RFID-Tags, können vor allem in der Industrie Warentransporte und Produkte während der Produktion zu jedem Zeitpunkt erfasst und ihre Bewegungen kontrolliert werden. Auch als Zugangskarten finden diese Tags ihre Verwendung, um z.B. bestimmte Bereiche nur für autorisiertes Personal zugänglich zu machen. Der Vorteil hierbei liegt in der kontaktlosen

Identifizierung des Karteninhabers. Somit muss kein zusätzlicher Code eingegeben werden und eine weitere Form der Authentifizierung fällt hierbei weg.

6.1.1 Grundlegender Aufbau

Prinzipiell besteht jedes RFID-System aus den folgenden Komponenten:

- Transponder
- Lese-, bzw. Schreib-/Lesegerät
- Antenne oder Antennenkombination
- Auswertungseinheit (Middleware)

Bei einem Transponder handelt es sich um einen Sammelbegriff für Datenträger mit einer Antenne und ggf. einem Datenspeicher. Diese werden berührungslos über eine bereitgestellte Luftschnittstelle eines Lesegerätes ausgelesen. Dabei gibt es zwei Transponderversionen: mit einer unveränderbaren Identifikationsnummer /Read-Only-Transponder) und mit einer veränderbaren Identifikationsnummer /Read-Write-Transponder. Des Weiteren werden zwischen batteriebetriebenen (aktiven Transpondern) und batterielosen (passive Transponder), welche vom magnetischen Feld bzw. von den elektromagnetischen Wellen des Schreibe-/Lesegerätes versorgt werden können. Dieses verfügt über eine Schnittstelle zu dem übergeordneten System (der Middleware). Hier werden alle erfassten Daten verarbeitet und bei Bedarf und Speichermöglichkeit des Transponders, wieder zurück zu diesem übertragen. Die Middleware steuert außerdem den Datenaustausch zu höheren Systemen, wie z.B. einem ERP(Enterprise Ressource Planning)-System. (Finger, 2008, S.4ff)

Im nachfolgenden sollen, einige Vor- und Nachteile der RFID-Technologie in Tabelle 9 zusammengefasst werden.

Vorteile	Nachteile
- Lesegerät muss nicht ausgerichtet werden - sekundenschnelle Identifikation der Objekte und dies eindeutig - teilweise sehr kleine Transponder - geringe Fehlerrate und große Lesegenauigkeit	- geringe Reichweite (bei kostengünstigen Systemen) - schlechte Standardisierung - Störanfällig durch Metall oder Flüssigkeiten - Datenschutz noch in der Entwicklung

Tabelle 9: Vorteile und Nachteile von RFID-Systemen
(Franke, 2006, S.72)

6.1.2 Frequenzen und Reichweite

RFID- Systeme unterscheiden sich vor allem in der Betriebsfrequenz des Lesegerätes, dem physikalische Kopplungsverfahren und der Gesamtreichweite des Systems. Das Frequenzband verläuft dabei von Langwellenbereich mit 135 kHz bis hin zum Mikrowellenbereich von 5,8 GHz. Und der Bereich der physikalischen Kopplung beinhaltet die Verwendung von elektrischen-, magnetischen- oder elektromagnetischen-Feldern. Des Weiteren variiert die Reichweite, je nach Anwendungsfall, zwischen 1mm und größer als 15m.

Systeme mit sehr kleinen Reichweiten werden Close-Coupling-Systeme genannt. Diese werden überwiegend in Bereichen mit großen Sicherheitsanforderungen eingesetzt, bei denen die Reichweite keine Rolle spielt. Dazu gehören z.B. elektronische Türschließanlagen oder kontaktlose Chipkatensysteme mit Zahlungsfunktionen. Sie verwenden sowohl elektrische als auch magnetische Felder zu Kopplung und werden mit Frequenzen bis zu 30 MHz betrieben.

Werden Reichweiten bis zu 1m benötigt, so werden die sog. Remote-Coupling-Systeme eingesetzt. Besonders weit verbreitet sind dabei die Systeme, welche die induktiven-Kopplungen verwendet, weswegen sie auch induktive Funkanlagen bezeichnet werden. Die Sendefrequenz dieser Systeme befindet sich dabei zwischen 135 kHz und 13,56 MHz.

RFID-Systemen mit einer Reichweitenanforderung von über 1m, werden als Long-range-Systemen bezeichnet. Diese werden im Ultra-Hoch-Frequenz- und Mikrowellenbereich betrieben und meistens aufgrund ihres physikalischen Funktionsprinzips als Backscatter-Systeme (Lesegerät sendet elektromagnetische Wellen aus) bezeichnet. Der Frequenzwellenbereich erstreckt sich dabei vom Ultra-Hoch-Frequenz-Bereich von 868 MHz bis zum Mikrowellenbereich von 5,8GHz.

Bei passiven (batterielosen) Backscatter-Transpondern können Reichweiten von 3m, bei aktiven (batteriebetriebenen), sogar Reichweiten bis zu 15m erreicht werden. Die Batterie wird dabei nicht zur Energieversorgung der Datenübertragung zwischen Transponder und Lesegerät benötigt, sondern zur Versorgung des Mikrochips und die Erhaltung der gespeicherten Daten. Zur Datenübertragung wird ausschließlich die Energie benutzt, welche vom Lesegerät ausgestrahlt und vom Transponder empfangen wird. (Finkezeller, 2008, S. 22ff)

6.1.3 Passive und aktive Transponder

Da passiven Transpondern keine eigene Energieversorgung besitzen, wird die gesamte benötigte Energie durch die eingebaute Antenne dem magnetischen oder elektromagnetischen Feld des Lesegerätes entnommen. Nur durch die vom Lesegerät bereitgestellten Felder kann eine Verbindung zwischen diesem und dem passiven Transponder aufgebaut werden. Dieser besitzt somit keinerlei Möglichkeiten, außerhalb der Reichweite eines Lesegerätes Signale auszusenden. (Finkezeller, 2008, S. 23f)

Bei aktiven Transpondern können eigene Energieversorgungen in Form von Batterien oder Solarzellen installiert werden und dienen dem Mikrochip als Energieversorgung. Durch diesen Umstand wird auch ein deutlich schwächeres Feld zum Betrieb benötigt, was sich positiv in der Reichweite auswirkt. Ähnlich dem passiven Transponder, ist es dem aktiven Transponder allerdings nicht möglich ein eigenes hochfrequentes Signal zu erzeugen. Die Datenübertragung ist erneut abhängig von den elektromagnetischen Wellen des Lesegeräts. (Finkezeller, 2008, S. 24)

6.1.4 Fazit Radio Frequency Identification

Bei der Recherche nach einem qualitativen gutem Bausatz zu Implementierung in das Gesamtsystem wurden verschiedene Developer Pakete betrachte. Der Fokus wurde dabei ausschließlich auf Systeme gelegt, welche Reichweiten von mind. 3-4m aufweisen. Bei einem einfachen RFID-System mit Leseeinheit, aktivem oder passivem Transponder, Antenne und allem weiteren benötigten Zubehör, beliefen sich die Kosten auf 500-1000€. Systeme, welche im Nahfeldbereich von 1mm-10cm arbeiten, kosten hingegen nur ein Bruchteil dessen und bewegen sich im Bereich von 30-100€.

Obwohl diese Technologie bei der eindeutigen Identifikation der Person einen enormen Vorteil aufweist, wurde eine Implementierung, aufgrund des zu hohen Kosteneinsatzes nicht länger in Betracht gezogen.

6.2 Identifikation mittels Bluetooth

Bluetooth wurde primär zur Vernetzung mobiler Geräte und als Kabelersatz zum Anschluss von Peripheriegeräten entwickelt. Dabei ist es vor allem auf portable oder mobile Anwendungen wie Tastaturen oder Headsets, ausgerichtet. Ein weiteres Einsatzgebiet ist die Synchronisation eines mobilen Endgerätes, PDA oder Handy, mit einem PC, um Kalenderdaten oder Kontakte abzugleichen. Bei einer Frequenz von 2,4GHz, arbeitet es auf dem gleichen Frequenzband wie WLAN kann aber durch ein ausgeklügeltes Frequenzsprungverfahren Störungen einfach ausgleichen. Dabei springt es 1600-mal pro Sekunde nach einer pseudozufälligen Reihenfolge zwischen den 79 verfügbaren Kanälen umher. Durch die Wahl der Leistungsklassen können zwischen 1 und 100mW Sendeleistungen erreicht werden, was zu einer Reichweite von 10 bis 100 Meter führt.
Unter anderem können bei den neueren Bluetooth Versionen die benutzbaren Kanäle nach bestimmten Kriterien, wie Empfangssignalstärke und Paketfehlerrate gefiltert werden. Dadurch ist es möglich Kanäle, welche den Mindestanforderungen nicht entsprechen, in der weiteren Frequenzsprung-Abfolge auszulassen.
Bluetooth-Geräte erhalten Werksseitig eine eindeutig identifizierbare Geräteadresse (MAC-Adresse), welche für jede Hardware spezifisch ist. Der Bluetooth-Standard definiert bei der Suche nach neuen Bluetooth-Geräten eine spezielle Suchprozedur. Bei dieser wird eine

bestimmte Bitfolge ausgesendet, die vom Gerät mit dem übermitteln der Geräteadresse beantwortet wird. Schließen sich nun mindestens zwei bis maximal 8 Geräte zusammen, bilden sie ein PicoNet, bei dem es genau einen Master gibt. Alle weiteren agieren als Slave. Der Master definiert aufgrund seiner Geräteadresse und einer internen Uhr die pseudozufällige Frequenzsprung-Reihenfolge. Diese wird von allen teilnehmenden Geräten gleichermaßen eingehalten, wenn vorher eine spezielle Anmeldeprozedur durchgeführt wurde. Ein paralleler Einsatz in mehreren PicoNets ist durch die pseudozufällige Frequenzsprung-Reihenfolge durchaus möglich, hat aber ein Absinken der Datenrate und der Verbindungsqualität zu Folge. Diese Netze sind auch im Bluetooth-Standard definiert und werden ScatterNets genannt. (Finger, 2008, S.30)

Die Vor- und Nachteile der Bluetooth-Technologie sind in Tabelle 10 zusammengefasst.

Vorteile	Nachteile
- relativ günstig im Vergleich zu anderen Systemen - schnell einsatzbereit - kein weiterer Installationsaufwand von zusätzlicher Hardware (vgl. WLAN) - gute Übertragungsraten (vgl. ZigBee)	- geringe Reichweite (vgl. WLAN) - begrenzte Anzahl an Netzteilnehmer - hoher Akkuverbrauch (vgl. ZigBee)

Tabelle 10: Vorteile und Nachteile von Bluetooth-Systemen

6.2.1 Grundlegender Aufbau eines Identifikationssystems

Analog zum RFID-Aufbau, besteht das Bluetooth-System aus folgenden Komponenten:
- Transponder (Bluetoothchip mit Antenne und Gehäuse)
- Schreib/Lesegerät (Bluetooth-Modul in Form eines USB-Dongles mit integrierter Antenne)
- Middleware-Programm (Schnittstellensoftware)

Wie auch beim RFID-System, wird hierbei das gleiche physikalische Grundprinzip, basierende auf elektromagnetischen Wellen, verwendet. Allerdings werden diese Wellen nicht nur empfangen, sondern der Transponder sendet selbst welche aus, was ihn zu einem aktiven Transponder macht. (Finger, 2008, S.41)

Bluetooth-Transponder

Bei dem Bluetooth-Transponder handelt es sich um ein autonomes Hardwaremodul, welches eine bestimmte Menge von Daten aufnehmen, speichern und abgeben kann. Damit dieser überhaupt gefunden und kontaktiert werden kann wird ein Bluetooth-Modul

als Schnittstelle zum Schreib-/Lesegerät eingesetzt. Die Versorgung des eingebauten Mikrocontrollers ermöglicht schließlich die eigene Energieversorgung des Bluetooth-Transponders. (Finger, 2008, S.42)

Schreib-/Lesegerät

Wie bereits beschrieben, interagiert das Schreib-/Lesegerät mit dem Transponder, da dieser nie selbständig eine Verbindung aufbaut. Dazu werden alle Bluetooth-Geräte in der Umgebung gesucht und identifiziert, um anschließend mit dem Transponder in Verbindung zu treten. Dadurch können die auf ihm gespeicherten Datensätze gelesen, erarbeiten und neu beschrieben werden. Analog zum Transponder, besteht das Schreib-/Lesegerät aus einer Bluetooth-Schnittstelle und programmierbarer Logik zur Steuerung und Datenverarbeitung. Durch die verbreitete Bluetooth-Technologie, kann programmierbare Bluetooth-Hardware eingesetzt werden, welche den Einsatz einer reinen Software-Lösung für das Schreib-/Lesegerät ermöglicht. Der Vorteil hierbei, besteht in der vereinfachten Entwicklung, der Flexibilität bei der Auswahl der Hardware und die Möglichkeit das System bei Bedarf zu erweitern. (Finger, 2008, S.42)

6.2.2 Versuche mit Bluetooth

Für die Versuche mit dem Bluetooth-System, wurde das Linux integrierte hcitool verwendet. Dieses hat verschiedene Testmöglichkeiten mit denen ein Bluetoothgerät konfiguriert werden kann. Zunächst werden einige Grundbegriffe wie Link Quality Indicator (LQI), Received Signal Strength Indicator (RSSI), und Transmit Power Level (TPL) erläutert, welche in der Testphase des Bluetooth-Systems als Messparameter benutzt wurden.

LQI

Der Link quality indicator (LQI) gibt die Qualität der empfangenen Datenpakete des Empfängers an und wird für jedes Packet extra berechnet. Dabei kann die Received Signal Strenght (RSS) verwendet werden, um den Grad der Signalqualität zu ermitteln. Dieser misst die vorhandene Gesamtenergie des empfangenen Signals. Der Signal-zu-Rausch-Grad (Englisch: signal-to-noise-ratio, SNR) ist eine andere Methode um die Signalqualität zu ermitteln. Als generelle Regel gilt: bei einem hoher SNR werden weniger Fehler in der Datenübertragung durchgeführt, weswegen ein hoher SNR als Indikator für hohe Signalqualität gilt. Die Linkqualität kann sowohl durch die Ermittlung der Signalenergie als auch durch den SNR ermittelt werden. (Farahani, 2008, S. 37)

RSSI

Ein relativ einfacher Weg einen LQI zu berechnen, ist es den received signal strength indicator (RSSI) als Grad der Link Qualität zu benutzen. Dieser misst die empfangene

Signalstärke und kann für jedes Datenpaket extra berechnet werden. (Farahani, 2008, S. 229)

TPL

Laut der Dokumentation des verwendeten hcitools, gibt der TPL den Grad der Konnektivität zu einem angeschlossenen Bluetoothgerät an. Dabei kann noch gewählt werden ob die Standardmäßige oder die Maximale Übertragungsstärke verwendet wird. (Linuxcommand, 2002)

Testphase

Der Testaufbau bestand aus einem Bluetoothfähigem Handy (LG P990), einem Laptop mit Bluetooth 3.0 und einer Messstrecke von 20 m. Nachdem die beiden Bluetoothgeräte miteinander verbunden waren, konnten die verschiedenen Messwerte ermittelt werden. Zunächst wurde die Messwerte bei der Startposition von 0m erfasst und in einer Tabelle übertragen. Dieser Schritt wurde jeweils in Meterschritten wiederholt, bis die gesamte Strecke von 10 m erfasst wurde. Anschließend wurde noch eine Messung bei Streckenabschnitt 20 m durchgeführt. Die Ergebnisse der beiden Testläufe sind im weiteren Verlauf zusammengefasst.

Versuch 1 – Abhängigkeit verschiedener Parameter im Vergleich zur Distanz

Beim ersten Versuch wurden die einzelnen Distanzpositionen nacheinander abgelaufen. Außerdem wurde eine gewisse Zeit auf diesen Positionen verweilt, um die Ergebnisse stabil ermitteln und in eine Tabelle übertragen zu können.

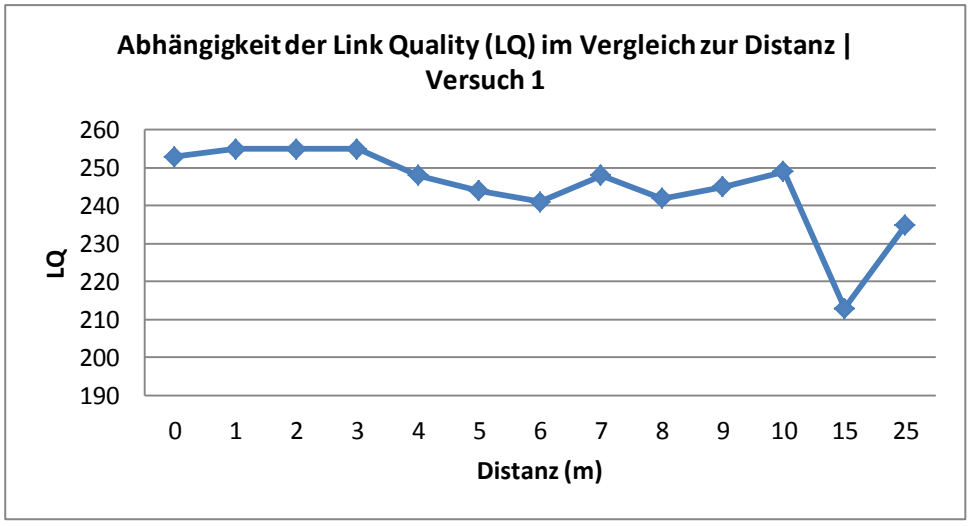

Abbildung 32: Abhängigkeit des LQ im Vergleich zur Distanz beim 1sten Versuch

Wie in Abbildung 32 zu sehen ist, verändert sich der LQ-Wert erst bei einer höheren Distanz merklich. Ein LQ-Wert von 255 entspricht dabei einer sehr guten Link Quality,

wohin gegen ein LQ-Wert von 0 einer sehr schlechten Link Quality entspricht. Diese Veränderung konnte allerdings nur zeitverzögert ermittelt werden. Wurde eine Distanzposition für eine längere Zeit eingehalten, so verbesserte sich der LQ-Wert wieder ein wenig. Dabei konnte eine Änderung des LQ-Wertes von 253 bei 0 m, auf einen LQ-Wert von 213 bei 15 m bzw. auf 235 bei 25m festgestellt werden.

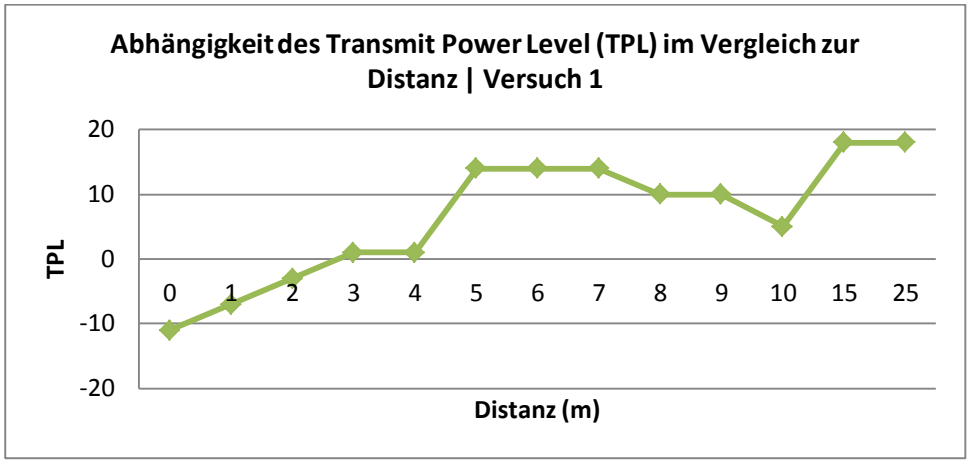

Abbildung 33: Abhängigkeit des TPL im Vergleich zur Distanz beim 1sten Versuch

Der TPL-Wert war in dieser Testreihe der aussagekräftigste Parameter. Hierbei konnte ein relativ guter Verlauf in Abhängigkeit zur Distanz ermittelt werden (Abbildung 33). Ein TPL-Wert von -11 entspricht dabei einer sehr guten Link Quality, wohin gegen ein TPL-Wert von +18 einem sehr schlechten Transmit Power Level entspricht. Der TPL-Wert veränderte sich von -11 bei 0 m, auf einen TPL-Wert von +18 bei 15 m bzw. 25 m. Die Änderungen erfolgten allerdings, ähnlich wie beim LQ-Wert, äußerst träge und erst nach längerem Stillstand an einer Distanzposition.

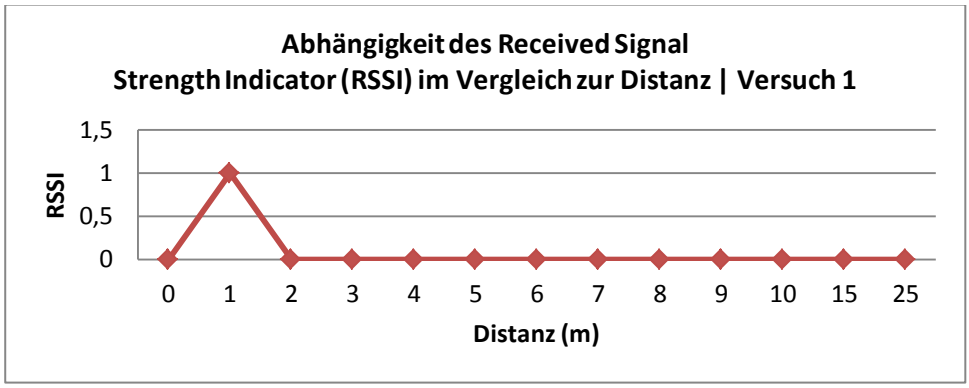

Abbildung 34: Abhängigkeit des RSSI im Vergleich zur Distanz beim 1sten Versuch

Die Messungen des RSSIs ergaben keine aussagekräftigen Ergebnisse (Abbildung 34). Dieser veränderte sich nur ein einziges Mal bei einer Distanz von 1 m auf einen RSSI-Wert

von 1. Bei allen weiteren Messungen wurde ein RSSI-Wert von 0 ermittelt. Somit kann dieser Parameter nicht für die Ermittlung einer Distanz verwendet werden.

Versuch 2 – Abhängigkeit verschiedener Parameter im Vergleich zur Distanz

Auch der Versuch 2 wurde nach den gleichen Kriterien wie Versuch 1 durchgeführt. Die Distanzpositionen wurden nacheinander abgelaufen. Auch die Verweildauer war ähnlich, um die Ergebnisse stabil zu erhalten und diese in eine Tabelle übertragen zu können.

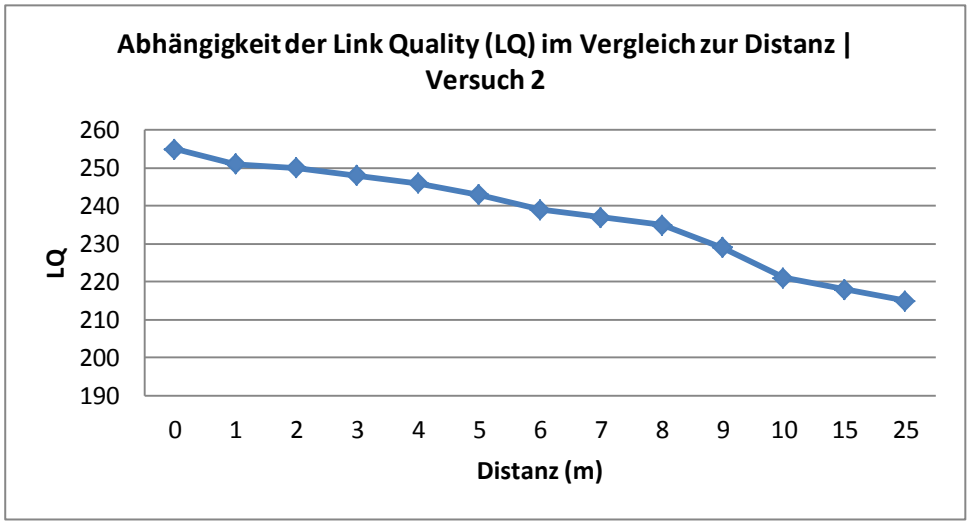

Abbildung 35: Abhängigkeit des LQ im Vergleich zur Distanz beim 2sten Versuch

Wie in Abbildung 35 zu sehen ist, verändert sich der LQ-Wert kontinuierlich nach unten. Dabei wurde eine Veränderung der Link Quality mit einem LQ-Wert von 255 bei 0 m, auf einen LQ-Wert von 215 bei 25 m ermittelt. Wie auch bereits beim vorherigen Versuch erwähnt, veränderten sich diese Werte äußerst träge.

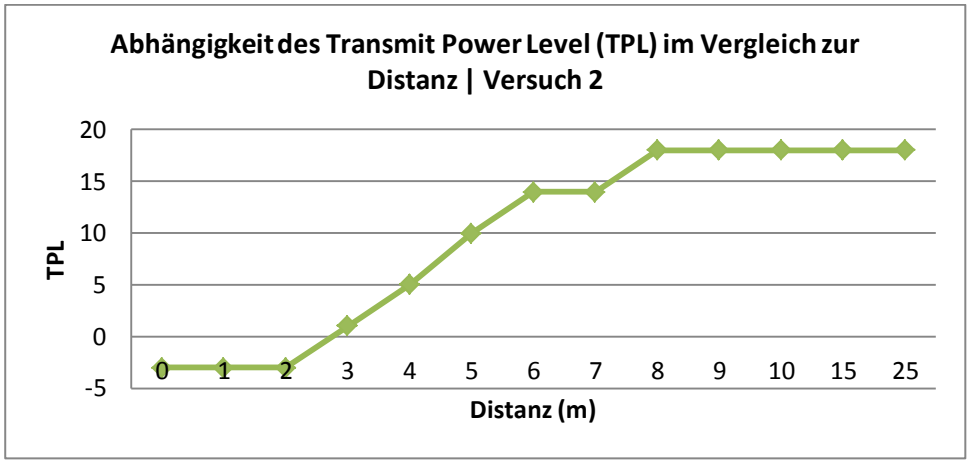

Abbildung 36: Abhängigkeit des TPL im Vergleich zur Distanz beim 2sten Versuch

Auch der TPL-Wert im Versuch 2 zeigt einen deutlichen Verlauf der Ergebnisse (Abbildung 36). Dabei ergibt sich ein Transmit Power Level mit einem TPL-Wert von -3 bei 0 m, bis hin zu einem TPL-Wert von +18 bei 25m, welcher allerdings schon bei einer Distanz von 8m erreicht wurde. Somit kann festgestellt werden, dass auch der TPL-Wert, zwar eine gewisse Aussage über die Abhängigkeit der Distanz gibt, dennoch nicht so stabil oder Echtzeitfähig ist, wie es eine Personenverfolgung benötigen würde.

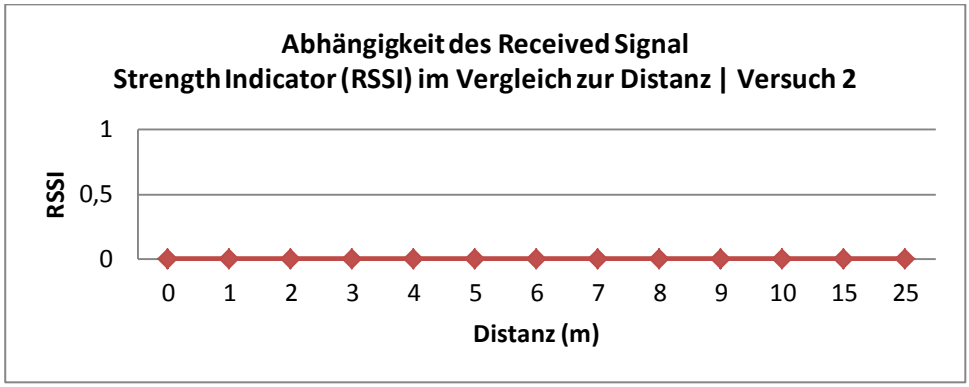

Abbildung 37: Abhängigkeit des RSSI im Vergleich zur Distanz beim 2sten Versuch

Im Versuch 2 konnte keinerlei Veränderung des RSSI-Wertes festgestellt werden (Abbildung 37). Wie bereits im vorherigen Versuch erläutert, gibt dieser Parameter keinerlei Informationen über die Abhängigkeit der Distanz zur Personenverfolgung.

Ergebnis der Versuche

Wie zu erkennen ist, wurden unterschiedliche Messwerte in den beiden Testläufen erfasst. Teilweise konnte zwar eine Veränderung der einzelnen Messparameter festgestellt werden, allerdings waren diese nicht aussagekräftig genug um eine Konsistenz zu erkennen. Besonders auffallend war die Tatsache, dass alle Messparameter miteinander interagieren und sich gegenseitig einstellen. So veränderte sich ein Messparameter während des zweiten Testlaufs nicht maßgeblich während die anderen Messparameter sich neu einzustellen schienen. Außerdem hatten die Messparameter eine gewisse zeitliche Verzögerung bei der Einstellung auf die neue Distanz, was das Ergebnis möglicherweise weiter verschlechterte.
Somit wurde der Schluss gezogen, dass ein Messparameter alleine, keinerlei Aussagekraft über die Entfernung einer Person hat. Die drei Messparameter in Kombination würden weiterer mathematischer Berechnungen bedürfen, um aussagekräftige Ergebnisse zu erhalten.

Ein dritter Testlauf wurde mit einer bestehenden Datenübertragung zwischen Mobiltelefon und Laptop durchgeführt und lieferte weitaus bessere Ergebnisse. Die Signalstärke während der Übertragung war stark abhängig von der Entfernung zum Laptop. Hierbei

konnten die Messwertveränderungen wesentlich schneller beobachtet werden. Um diese Ergebnisse zu erhalten ist allerdings eine konstante Datenübertragung zwischen den beiden Geräten nötig, was wiederum zum erhöhten Batterieverbrauch führt. Somit führte auch dieser dritte Versuch zu keinem praktikablen Ergebnis.

6.2.3 Fazit

Als kostengünstigste Alternative im Vergleich zu den anderen Systemen, konnte das Bluetooth-System als einziges intensiv getestet werden. Die Ergebnisse waren nicht aussagekräftig genug um eine vernünftige Positionsangabe durchführen zu können. Allerdings konnte relativ gut ermittelt werden, ob sich eine Person nun vom Roboter weg oder zu ihm hin bewegt.

6.3 Identifikation mittels WLAN

Ein Wireless-Local-Area-Network (WLAN) ist ein lokales Funknetz nach dem IEEE 802.11 Standard, welches auf einem Frequenzband zwischen 2,4GHz und 5GHz. Die Reichweite kann dabei bis zu 100 Metern betragen und wird durch verschiedene Materialien stark beeinflusst werden. In einem WLAN-Netz ist es durchaus möglich eine Lokalisation bis auf 1-2 m genau durchzuführen. Allerdings ist hierbei erneut ein komplettes Netzwerk von sog. Knoten nötig um eine stabile Positionsvorhersage treffen zu können. Hierbei sendet ein Access Point kleine Datenpakete (Beacons) in Intervallen von ca. 10 Paketen pro Sekunde (10Hz) an alle Knoten im Sendebereich. In diesen Beacons ist vor allem die Feldstärke des Signals gespeichert, welche anschließend dazu dient die Position in einem Raum zu bestimmen. Dazu wird davon ausgegangen, dass die Feldstärke an bestimmten Punkten im Raum gleich bleibt und diese Information wird mit der gespeicherten und der aktuell gemessenen Feldstärke abgeglichen. Alternativ dazu kann über das Laterationsverfahren (Positionsermittlung über Distanzberechnungen) mit zwei Accesspoints die Zeit zwischen Absenden eines Signals und Eintreffen der Antwort erfasst und daraus die Position zum Endgerät ermittelt werden. (Linzner, 2010, S. 21)
Die Vor- und Nachteile der WLAN-Technologie sind in nachfolgender Tabelle 11 zusammengefasst.

Vorteile	Nachteile
- WLAN ist mittlerweile auch Städteweit verfügbar - Lokalisation bereits mit einfachen, quelloffenen Systemen realisierbar	- Für Navigation und Lokalisation in Räumen noch zu ungenau - Netzwerke und Accesspoints müssen erst aufgebaut werden

Tabelle 11: Vor- und Nachteile des WLAN-Systems

Fazit

Wireless-LAN ist als mobiler Internetzugang nicht mehr weg zu denken. Aufgrund der gestiegenen Mobilität und erweiterten Möglichkeiten hat es in vielerlei Hinsicht das kabelgebundene LAN weitgehend abgelöst. Als Navigation oder Lokalisation von mobilen Robotern ist es allerdings noch viel zu ungenau. Der zusätzliche Installationsaufwand von Access Points und Routern macht eine schnelle und einfache Verwendung, wie es z.B. beim Bluetooth-System der Fall ist, nicht möglich.

6.4 Identifikation mittels ZigBee

Das ZigBee-System wurde als Funktechnologie zur Datenübertragung über kurze Distanzen von 30 bis 100m entwickelt. Dabei werden Frequenzbereiche von typischerweise 868 MHz bis 2,4 GHz, bei einer maximalen Datenrate von 250 kBit/s verwendet. Zunächst wurde es für die industrielle Anwendung, z.B. Überwachungen und Positionsbestimmungen von Gütern vorgesehen, entwickelte sich allerdings schnell weiter zu einem Einsatz als Wireless Personal Area Network (WPAN), welches dem Einsatz von Bluetooth nahe kommt. Aufgrund der geringen Datenrate, kann ZigBee allerdings nicht als direkter Konkurrent von Bluetooth angesehen werden. Dieses System zielt vor allem auf den äußerst geringen Stromverbrauch der einzelnen Komponenten ab, weswegen es in mobilen Anwendungen breite Akzeptanz genießt. Durch das spezielle ZigBee Protokoll, lassen sich theoretisch 216 Knoten (65.000 Teilnehmer) realisieren. Dabei können diese automatisch und dynamisch aufgebaut werden, so dass ein Signal auf einem Zick-Zack-Kurs seinen Weg durch das Netz nimmt. Aufgrund dieser Verhaltensweise, welcher dem Flug einer Biene gleichkommt, wurde der Name dieser Technologie festgelegt. (Dembowski, 2009, S.378)

Tabelle 12 schlüsselt hierbei die Vor- und Nachteile des ZigBee-Systems auf.

Vorteile	Nachteile im Vergleich zu Bluetooth
- Niedriger Stromverbrauch - Einfache Implementierung in die Endgeräte - Kleine, kompakte Bauweise der Funkmodule - Geringe Reaktionszeiten	- Geringe Bandbreite (max. 300 kBit/s) - Geringere Übertragungssicherheit - Geringere Funknetzstabilität (da keine Frequenzsprünge) - Hoher Aufwand, um ausfallsichere Netze herzustellen - Hohe Anschaffungskosten der einzelnen Module

Tabelle 12: Vor- und Nachteile des ZigBee-Systems
(nach Sorex, 2008)

6.4.1 Versuch mit ZigBee

Wie bereits im Kapitel 6.2.2 erläutert, gibt der Link Quality Indicator (LQI) die Qualität der empfangenen Datenpakete des Empfängers an. Forscher der Universität von Rostock publizierten im Juli 2007 ihre Ergebnisse bei der Verwendung des LQIs zur Distanzberechnung (Grossmann et al., 2007, S. 5) Dabei handelte es sich um einen Testaufbau mit 2 ZigBee Knoten und einem PC, welcher die Daten speichert. Ein Knoten fungierte als Referenzknoten (Beacon) und sendete permanent Datenpakete. Der zweite Knoten speicherte den LQI der ankommenden Datenpakete und übermittelte diesen an den PC. Während der Testphase wurde die Position des Beacons zwischen 0 m und 40 m variiert und dies insgesamt 20mal wiederholt. Außerdem wurden alle Messungen mit 4 verschiedenen ZigBee-Modulen durchgeführt.
Aus den Ergebnissen konnte eindeutig geschlussfolgert werden, dass eine Vergrößerung der Distanz, eine Verkleinerung des LQIs zur Folge hat. Die Messwerte fielen von 150byte/LQI bei 1 m auf 20-30byte/LQI bei 40m. (Grossmann et al., 2007, S. 6)

6.4.2 Fazit

Obwohl ZigBee ein durchaus vielversprechendes System zur Implementierung in das Gesamtsystem des Projektes war, wurde eine Umsetzung aufgrund der relativ hohen Anschaffungskosten eines Entwicklerpaketes nicht durchgeführt. Bei der Recherche nach einer entsprechenden Hardware konnten nur Systeme ab 200€ ermittelt werden. Aufgrund der Tatsache, dass ein ZigBee-System, lediglich die Distanz zu einer Person, aber nicht die genaue Position dieser ermitteln kann, wurden die Kosten als zu hoch betrachtet. Nichtsdestotrotz ist das ZigBee-System vor allem im mobilen Bereich, aufgrund seiner energiesparenden Hardware, durchaus Interessant für größere Folgeprojekte mit einer Vertiefung in diese Technologie.

6.5 Vergleiche zwischen den Funksystemen

Im weiteren Verlauf sollen einige Vergleiche zwischen den einzelnen Systemen durchgeführt werden. Dabei werden vor allem Vergleiche mit dem Bluetooth-System gezogen, welches als einziges ausführlich getestet werden konnte.

6.5.1 Vergleich zwischen Bluetooth und ZigBee

Ähnlich wie Bluetooth ist ZigBee ein WPAN-Netzwerk, welches immer zur Verfügung steht. Die erzielbaren Datenraten sind zwar geringer, allerdings wird dieser Nachteil, durch den äußerst geringen Stromverbrauch wieder aufgehoben. Dadurch, dass der „Hibernation-Mode“ bereits im Protokoll-Stack implementiert ist, kommen diese Systeme mit nur einer Batterie, Monate bis Jahre aus. Auch ist ein stetiger Preisfall im Vergleich zu Bluetooth-Systemen zu beobachten. Ein weiterer Vorteil liegt darin, dass sich ZigBee-Geräte selbständig organisieren und somit vermaschte Netzwerke aufbauen. Bluetooth hingegen,

wurde ursprünglich als Ersatz von Telefon, Computer und Headsets entwickelt, weswegen ein regelmäßiges Aufladen vorausgesetzt wird.
Tabelle 13 zeigt einen Vergleich dieser beiden Systeme an, wobei die relative Einschaltdauer mit „Duty Cycles“ bezeichnet wird. (Gessler, Krause, 2009, S.255ff)

Parameter	ZigBee	Bluetooth
Datenpakete	kleine Datenpakete über relativ große Netzwerke	große Datenpakete über relativ kleine Netzwerke
Netzwerke	überwiegend statisch mit vielen Knoten	wenige Knoten bei Ad-hoc Netzwerken
Netzwerk-Verbindung	schnell – typisch 30ms	langsam – typisch > 3s
Slave-Aktivierung (schlafen->aktiv)	typisch 15ms	typisch 3s
Zugriffszeit auf aktiven Slave	15ms	2ms
Resume	kleine bis sehr kleine „Duty cycles“ bei statischen und dynamischen Umgebungen mit vielen aktiven Knoten	hoher Quality of Service. Begrenzte Knotenzahl mit moderaten Datenraten

Tabelle 13: Vergleich zwischen ZigBee und Bluetooth
(nach Gessler, Krause, 2009, S.258)

6.5.2 Vergleich zwischen Bluetooth und WLAN

In diesem Fall arbeiten beide Funkstandards im gleichen Frequenzband von 2.4GHz und können unterschiedliche Geräte über Funk miteinander verbinden. Da Wireless LAN dem Bluetooth-System in punkto Reichweite weit überlegen ist, kommt es überwiegend in lokalen Netzwerken zum Einsatz. Außerdem werden hierfür schon Verbesserungen im industriellen Bereich angeboten. Dies ist vor allem der Hauptunterschied, zwischen diesen beiden Systemen. Bluetooth war nie als Netzwerkprotokoll gedacht, sondern überwiegend als Kabelersatz für Peripherie Geräte, wie eine Tastatur, oder Computermäuse. Durch ihren geringen Stromverbrauch haben diese allerdings auch eine geringere Reichweite, was sich allerdings wieder im geringeren Preis niederschlägt. Ein weiterer Vorteil von Bluetooth ist die gleichzeitige Verwendung der beiden Übertragungsverfahren SCO (Synchronous Connection Oriented Link) und ACL (Asynchronous Connectionless Link). SCO dient dabei zur Audioübertragung, z.B. bei einem Telefonat, wohingegen ACL die 100%tige Übertragung aller gesendeten Datenpakete garantiert. Dadurch kann ein äußerst hoher Quality of Service erzielt werden. (Gessler, Krause, 2009, S.259f)

6.5.3 Vorteile von Bluetooth gegenüber RFID

Ein erster Vorteil der Bluetooth-Technologie, gegenüber dem RFID-System, findet sich in der bereits bestehenden Standardisierung der Kommunikation zwischen den einzelnen Komponenten. Diese sind in der Bluetooth-Spezifikation festgelegt, können somit leicht implementiert werden und sichern eine weltweite Kompatibilität zwischen den Geräten.
Besonders die Dimensionen der Schreib-/Lesegeräte spielt bei mobilen Anwendungen eine große Rolle. Diese sind teilweise so klein, dass sie als PC-Karte oder USB-Dongle, ohne weitere Verkabelung an den PC, angeschlossen werden können. Die äquivalenten RFID-Geräte sind im Vergleich dazu um ein vielfaches Größer und benötigen zudem noch eine Verkabelung der Reader und der Antennen.
Die Kosten für beide Systeme unterscheiden sich zudem erheblich voneinander. Hierbei stehen 25€ für ein USB-Dongle mit Antenne, 2300€ für ein HF-Reader und eine Einzelantenne gegenüber. Tabelle 14 schlüsselt nochmals den Preisunterschied auf.

Produkt	Kosten [€]
HF-Reader mit Multiplexer für mehrere Antennen	ca. 2000,-
HF-Einzelantenne	ca. 300,-
HF-Gate	ca. 2000,-
UHF-Reader mit Multiplexer für mehrere Antennen	ca. 4000,-
UHF-Einzelantenne	ca. 400,-
Bluetooth-USB-Dongle	ca. 25,-
Bluetooth-PC-Karte	ca. 60,-

Tabelle 14: Preisvergleich zwischen Bluetooth- und RFID-Komponenten
(nach Finger, 2008, S.37)

Bei einer Sendeleistung von 100mW können Bluetooth-Module über eine Entfernung von bis zu 100m, Daten übertragen. Dies geht weit über die Möglichkeiten der meisten aktiven RFID-Systeme hinaus. Des Weiteren sind sie nicht so störanfällig gegenüber metallischen Untergründen und haben ein höheres Durchdringungsvermögen bei den meisten Stoffen.

6.5.4 Nachteile von Bluetooth gegenüber RFID

Beim direkten Vergleich zwischen diesen beiden Systemen, kommt vor allem der Faktor der Energieversorgung zum Tragen. Dieser ist der wesentliche Nachteil der Bluetooth-Technologie, gegenüber dem RFID-System (bei Verwendung von passiven Transpondern). Durch die primäre Entwicklung für Peripheriegeräte und solche, welche öfters aufgeladen werden können, wurde der Fokus in erster Linie, nicht auf den batterieschonenden Betrieb gelegt, weswegen sie für die reine Identifikationsaufgabe

weitaus überdimensioniert sind und zusätzliche Anpassungen benötigen. Des Weiteren verdeutlicht Tabelle 15 die Größen- und Gewichtunterschiede zwischen den beiden Systemen.

Transponder	Abmessungen	Gewicht [g]
RFID-LF-Coin passiv	16 x 3	1,2
RFID-HF-Label passiv	80 x 50 x 1	1
RFID-UHF aktiv	131 x 28 x 21	50
Bluetooth Transponder Gehäuse mit 3x AAA Batterien	67 (87 mit Halterung) x 42 x 28	89

Tabelle 15: Größenvergleich zwischen Bluetooth-Transponder und aktiven, passiven RFID-Transpondern
(nach Finger, 2008, S.37)

6.6 Zusammenfassung der Ergebnisse

Zusammenfassend betrachtet haben alle Funktechnologien durchaus ein großes Potential zur Positionsermittlung einer Person in einem Raum. Dabei brauchen manche Systeme mehr Hardware als andere und bei einigen ist der zusätzliche Installationsaufwand nicht zu vernachlässigen. Auch wenn die Transpondermodule beim RFID-System relativ kostengünstig sind, verhinderten die teilweise extrem hohen Kosten für das RFID-Lesegerät (+ Antenne) eine intensivere Testreihe. Die relative Ähnlichkeit zwischen dem ZigBee- und Bluetooth-System führten schließlich dazu, dass keine ZigBee-Entwicklerplattform gekauft wurde. Die Tatsache, dass Bluetooth bereits im Mobiltelefon und Laptop verbaut waren verstärkte diese Entscheidung nochmal zusätzlich. Dadurch konnte die Testreihe zu Kosten von 0€ durchgeführt werden.
Die durchgeführten Tests ergaben ein teilweise akzeptables Ergebnis. Zur besseren und schnelleren Distanzermittlung musste aber eine kontinuierliche Datenübertragung stattfinden, welche sehr zur Lasten der Akkuleistung ging. Die erzielten Ergebnisse hingen auch stark von der verwendeten Hardware ab. Bei den durchgeführten Tests, wurden bereits installierte kostengünstige Bluetoothgeräte verwendet. Mit sorgfältig ausgewählter und abgestimmter Hardware, welche einen besseren Zugriff auf die einzelnen Sendeparameter ermöglicht, wären die Ergebnisse sicherlich besser und aussagekräftiger ausgefallen. Aus diesem Grund und aufgrund der gesammelten Erkenntnisse, wurde keines der Funksysteme in das Gesamtsystem eingesetzt.

7 Robotersteuerung

Um die Bewegungen des Roboters effektiv steuern zu können, wird auf die Funktionalitäten im ROS zurückgegriffen. Im weiteren Verlauf dieses Kapitels, werden die verschiedenen Methoden aufgezeigt, einen physikalischen Roboter oder, wenn dieser nicht zur Verfügung steht, eine Robotersimulation, entsprechend der Programmierung zu steuern. Die Erkenntnisse, welche hierbei gewonnen werden, dienen schließlich den beiden Versionen des Gesamtsystems, als Grundlage bei der Steuerung des Roboters.

7.1.1 Steuerung des Roboters

Parallel zu den Hauptprogrammen werden noch zusätzliche Programme in einer Konsolenanwendung gestartet, welche dazu dienen verschiedene Treiberpakete, Nodes oder Topics zu laden. Diese werden für den Programmablauf und für die Steuerung des Roboters benötigt und wurden teilweise entsprechend den Anforderungen modifiziert und bearbeitet.
Dabei handelt es sich um folgende Programme bzw. Treiber:

- Joy, Node/Treiber zur Steuerung des Controllers
- p2os_driver, Treiber zur Steuerung des Roboters
- p2os_dashboard, Zusatzprogramm zur Steuerung und Visualisierung div. Roboterparameter

Und optional:

- stage, einfache Robotersimulationsumgebung

Ein Roboter mit einer ROS-Installation subscribed automatisch zu einem Topic namens „cmd-vel“. Über diesen Knoten werden die wichtigen Steuerbefehle gesendet und hierbei wird auch definiert, in welche Richtung er sich mit welcher Geschwindigkeit bewegen soll. Dabei können Bewegungen in X-Richtung (also vorwärts, rückwärts) oder auch in eine theta-Richtung (also Rotation um die eigene Achse) übermittelt werden. Bewegungen in Y-Richtung bleiben dabei unberücksichtigt, da sie nur für holonome Fahrzeuge oder Flugroboter relevant sind.

ROS verfügt auch über eine Robotersimulation (Stage, Abbildung 38-rechts), welche überaus hilfreich ist, wenn der physikalische Roboter nicht zur Verfügung steht oder nicht jede Programmänderung direkt an ihm getestet werden sollte. In diesem wurden auch die ersten Tests der Steuerung durchgeführt, da noch nicht gewährleistet werden konnte, dass sich der Roboter auch wie erwartet verhält. Stage hört auch auf die cmd-vel-Befehle, welche von einem Programm geschickt werden können und bewegt sich ähnlich dem physikalischen Roboter.

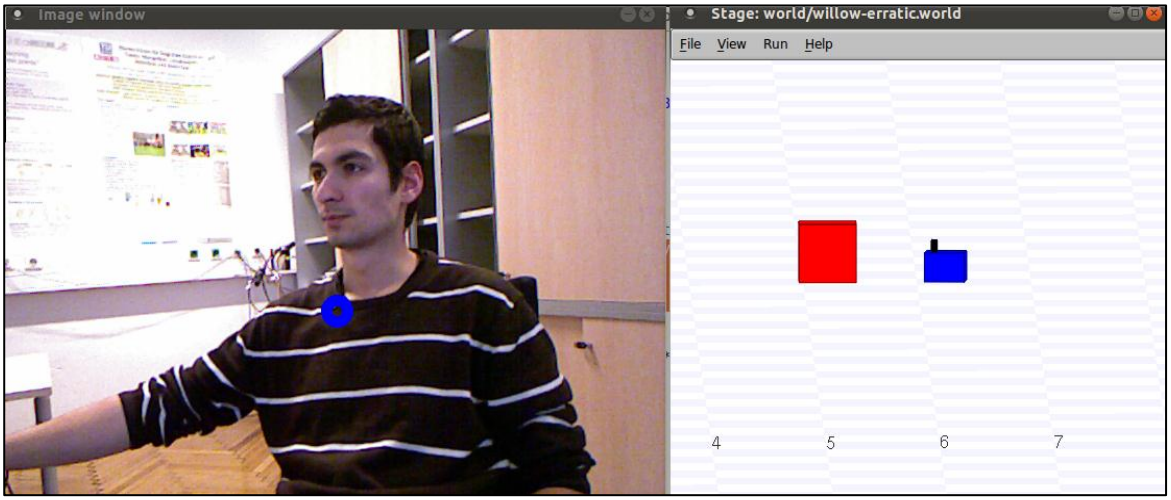

Abbildung 38: Detektierte Person (links) und Simulation des Roboters (rechts)

Bevor das Programm am Roboter getestet werden konnte musste zunächst noch eine Sicherheit eingebaut werden, falls dieser außerplanmäßige Bewegungen durchführt. Mithilfe eines Wireless-Controllers einer PlayStation 3 und dem entsprechenden ROS-Package können nun über diesen direkte Befehle an den Roboter geschickt werden. Somit wird gewährleistet, dass er auch im Falle von auftretenden Problemen sicher zum Stillstand kommt. Außerdem kann der Roboter sich nur bewegen, wenn ein bestimmter Knopf am Controller gedrückt wird. Wird dieser losgelassen, bleibt dieser wieder stehen und führt keine weiteren Befehle mehr bis zum erneuten Drücken des Knopfes aus. Hierbei wurde eine zusätzliche Codezeile in den Treiber des Controllers integriert, um bestimmte Buttons mit entsprechenden Befehlen versehen zu können.
Die Kommunikation zum physikalischen Roboter geschieht hierbei über ein USB-zu-RS232 Adapter welcher über einen USB-Hub mit dem Laptop verbunden ist.

7.1.2 Vorderen Hindernissen erkennen und ausweichen

Ohne weitere Informationen über Hindernisse oder Objekten vor dem Roboter, hat er keinerlei Ansatzpunkte diesen auszuweichen. Er würde unbeirrt versuchen durch diese hindurchzufahren, was unweigerlich in einer Kollision enden würde. Demzufolge sollten ihm die Installierten Sensoren bei der Kollisionsvermeidung behilflich sein. Einer dieser Sensoren ist der Hokuyo Laserscanner, welcher vorne am Roboter befestigt ist. Die Spannungsversorgung des Sensors erfolgt dabei über den Roboter selbst. Ein USB-Anschluss ermöglicht es außerdem den Laserscanner zu konfigurieren und die ermittelten Daten zum Laptop zu übertragen.

Einsatz des Hokuyo Laserscanners

Wie bereits im Kapitel 2.2.4 erläutert kann der Hokuyo Laserscanner er relativ genau Hindernisse in einer Ebene erkennen. Obwohl dieser Laserscanner nicht ins Gesamtsystem mitaufgenommen wurde, soll an dieser Stelle auf eine mögliche

Implementierung eingegangen werden. Dazu wurden verschiedene Tests durchgeführt welche aufgrund des Zeitmangels nicht weiter intensiviert werden konnten, um eine Verbindung mit dem Gesamtsystem zu ermöglichen.

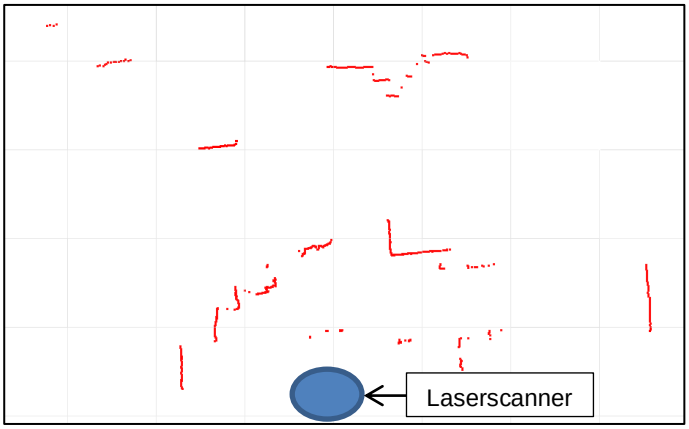

Abbildung 39: Erkennungsradius des Hokuyo Laserscanners

Wie auf Abbildung 39 zu sehen ist, erkennt der Laserscanner alle Hindernisse in seiner Umgebung, bis zu einem Abstand von ca. 6 m. Nun gilt es, diese Informationen für zu Filtern und dem Roboter über die bereits erwähnten cmd_vel-Befehle Bewegungsmuster vorzugeben.
Da der Lasertreiber bereits im ROS integriert ist, erstellt er bei der Initialisierung automatisch einen Knoten. Über diesen können alle empfangenen Sensordaten abgegriffen und verarbeitet werden. Dabei werden zunächst alle Laserbeams mit einem Wert von 0 herausgefiltert, da diese ungültigen Signalen entsprechen (unendlicher Abstand, keine Reflexion erhalten).

Von allen Laserbeams wird der mit dem kleinsten Wert ausgewählt. Dieser gibt die geringste Distanz eines Objektes zum Roboter an. Anhand von gewissen auswahlverfahren wird der Roboter nun entweder nach links oder rechts gedreht um eine Kollision zu vermeiden.
Diese simple Funktionsweise ermöglicht es, den Roboter selbstständig Hindernissen ausweichen zu lassen. Da die Implementierung des Hokuyo Laserscanners nur ein optionales Projektziel ist und eine tiefere Verknüpfung mit dem Gesamtsystem zu viel Zeit in Anspruch genommen hätte, wurde dieses Ziel nicht weiter verfolgt.

7.1.3 Hinteren Hindernissen erkennen und ausweichen

Um auch Hindernissen beim Rückwärtsfahren ausweichen zu können, wurden verschiedene Sensoren in Betracht gezogen. Diese würden allerdings zusätzliche Kosten (z.B. Hokuyo Laserscanner = 2-3000€) bedeuten. Bei der ersten verwendeten Roboterversion waren bereits zwei Bumper installiert. Diese konnten über einen

entsprechenden Treiber über ROS angesprochen und ausgelesen werden. Wie bereits in Kapitel 2.2.4 erläutert, würden Bumper nur eine Kollision erkennen, nicht aber vermeiden können. Aufgrund des Zeitmangels wurden aber keinerlei Umsetzungsversuche zur hinteren Kollisionsvermeidung durchgeführt. Hinzu kam der Wechsel des Roboters, welcher nicht mehr über die installierten Bumper verfügte. Eine Kollisionsvermeidung steht demnach weiterhin aus und kann allerdings durchaus in Folgeprojekten weiter verfolgt werden.

8 Umsetzung des Prototypen

Basierend auf den Ergebnissen aus dem theoretischen Kapiteln, kann nun das erworbene Wissen in die Umsetzung des Prototypens einfließen. Die nachfolgenden Kapitel sind nach ihrem Erscheinen im Programmablauf gegliedert.

Die Masterarbeit begann mit der Entwicklung der Version 1 des Personenverfolgungsprogramms auf Basis der 3D-Punktwolke der Kinect Kamera und des FastTemplateMatchers. Bereits während der Entwicklung, zeichnet sich die enorme aufzubringende Rechenleistung und zunehmend langsame Verfolgungsgeschwindigkeit ab. Die Alternative dazu wurde in einer Kombination aus dem Tiefenbild der Kinect Kamera und dem Particle Filter, basierend auf verschiedenen Merkmalen, gefunden. Aus dieser Alternative, entwickelte sich die weitaus schnellere Version 2 des Personenverfolgungsprogramms.

Im weiteren Verlauf dieses Kapitels werden beide Versionen und deren Aufbau im Detail erklärt, um anschließend miteinander verglichen zu werden. Eine Evaluation der gewonnenen Ergebnisse schließt das Kapitel ab. Abbildung 40 zeigt dabei den aktuellen Aufbau des Prototypens.

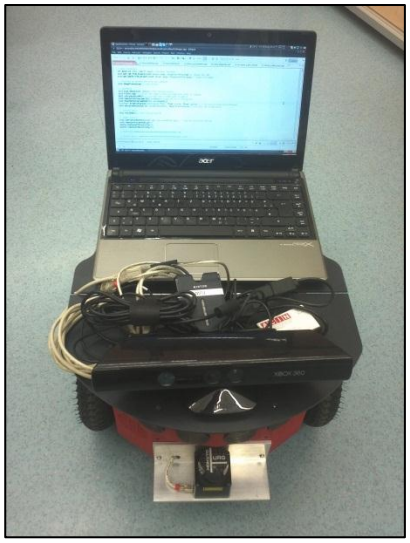

Abbildung 40: Aufbau des Prototypens

8.1 Version 1 – 3D-Punktwolke + FastTemplateMatcher

In diesem Kapitel soll die erste Version des RobotaS-Systems dargestellt werden. Diese entstand in der Anfangsphase des Projektes und basierte noch auf der 3D-Punktwolke der Kinect Kamera und den Filter- und Bearbeitungsfunktionen, die durch die Point Cloud Library implementiert werden können. Abbildung 41 zeigt ein Ablaufdiagramm, in welchem ersichtlich ist, wie die einzelnen Komponenten für die Durchführung des Programms miteinander verkettet und welche Funktionen aus dem Grundlagenkapitel dieser Arbeit in

das System eingeflossen sind. Bereits in anderen Kapiteln erläuterte Funktionen werden hierbei mit dem entsprechenden Kapitelhinweis erwähnt, aber die genaue Funktionsweise nicht mehr im Detail erläutert.

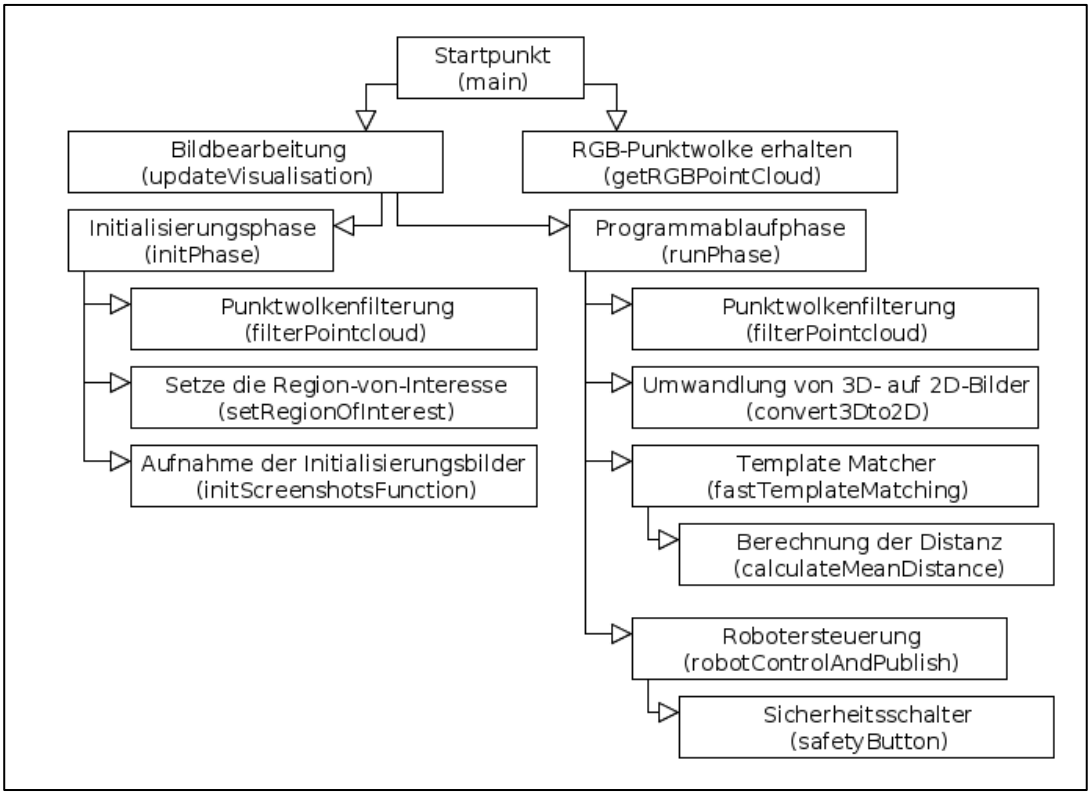

Abbildung 41: Ablaufdiagramm der Version 1 des Gesamtsystems

8.1.1 Startpunkt (main)

Ausgehend von der Initialisierung in der Main Funktion wird über den RGB_PointCloud_Knoten der Kinect Kamera eine 3D-Punktwolke mit RGB-Farbkodierung empfangen. Diese steht im weiteren Verlauf für die gesamte Bearbeitung dem System zur Verfügung. Außerdem wird hieraus auch zyklisch die updateVisualization-Funktion aufgerufen.

8.1.2 Bildverarbeitung (updateVisualisation)

Die updateVisualisation-Funktion dient dem Programm als Koordinator für alle weiteren Schritte. In dieser wird immer wieder eine neue 3D-Punktwolke empfangen, sowie die InitPhase und die RunPhase gestartet.

8.1.3 Initialisierungsphase (initPhase)

Die InitPhase dient dazu, die ersten Schritte bei der Erfassung der Person einzuleiten. In ihr werden die Funktionen filterPointCloud, setRegionOfInterest und

initScreenshotFunction gestartet. Wurden diese Funktionen erfolgreich ausgeführt, wird die Initialisierungsphase beendet und die RunPhase kann beginnen.

8.1.4 Punktwolkenfilterung (filterPointCloud)

Bei der filterPointCloud-Funktion wird die empfangene Punktwolke durch den Einsatz des Passthrough-Filters (Kapitel 4.2.2) zunächst auf einen gewissen Bereich verkleinert. Der entsprechende Code dafür sieht wie folgt aus:

```
//  filter x cloud_xyz_rgb
pass.setFilterLimits(-0.5, 0.5);
//  filter z cloud_xyz_rgb
pass.setFilterLimits(0.4, 1.4);
```

In diesem Fall wird die Punktwolke ab einem Abstand von 1,4 m und alles was sich weiter als 0.5 m, ausgehend von der Bildmitte links und rechts von dieser befindet abgeschnitten. Die Filterung verringert, wie bereits im Kapitel 4.2.2 beschrieben, die Menge der Punkte. Dadurch werden benötigte Berechnungen beschleunigt und es können mehr Informationen in kürzerer Zeit verarbeitet werden.

8.1.5 Region-von-Interesse (setRegionOfInterest)

Wie bereits im Kapitel 3.2.5 erläutert, kann ein Region-of-Interest (ROI) in einem Bild definiert werden. Dies wird auch in dieser Phase des Programmes durchgeführt. Von Interesse ist in diesem Fall nur der Inhalt, welcher sich, ausgehend von der Bildmitte, in einem definierten Bereich befindet. Dieser ROI dient der nachfolgenden initScreenshot-Funktion bei der Erstellung der Vorlagen für den FastTemplateMatcher.

8.1.6 Initialisierungsbilder (initScreenshot)

Die initScreenshot-Funktion übernimmt hierbei die Aufgabe, die Initialisierungsaufnahmen zu machen. Diese Vorlagen werden zur Detektion der Person vom FastTemplateMatcher (Kapitel 3.2.7) benötigt. Das Problem, welches sich hierbei stellt, ist die Sicht auf die Person zum Zeitpunkt der Bewegung. Wird eine Vorlage von der Frontansicht einer Person erstellt, kann sie zu einem gewissen Grad stabil genug wiedergefunden werden, wenn sie wieder frontal zur Kamera steht. Dreht sich die Person nun um und es ist nur noch der Rücken zu sehen, so kann es passieren, dass der FastTemplateMatcher keine Übereinstimmung mehr mit der Vorlage finden kann, da diese von der Frontseite der Person erstellt wurde. Um die Bildverarbeitung zu beschleunigen und robuster zu machen, werden zunächst 20 Vorlagen erstellt, wobei sich die Person einmal drehen muss. Somit wird gewährleistet, dass diese nun zu 360° erfasst wurde. Verliert der FastTemplateMatcher einmal sein Ziel, weil sich die Person evtl. gedreht hat, stehen ihm nun 20 Vorlagen zur Verfügung, welche er nacheinander mit dem aktuellen Bild vergleichen kann. Tests haben ergeben, dass sich dadurch die Erkennungsrate um ein

vielfaches steigern und beschleunigen lässt, zu einem relativ geringen Mehraufwand in der Berechnungsdauer.

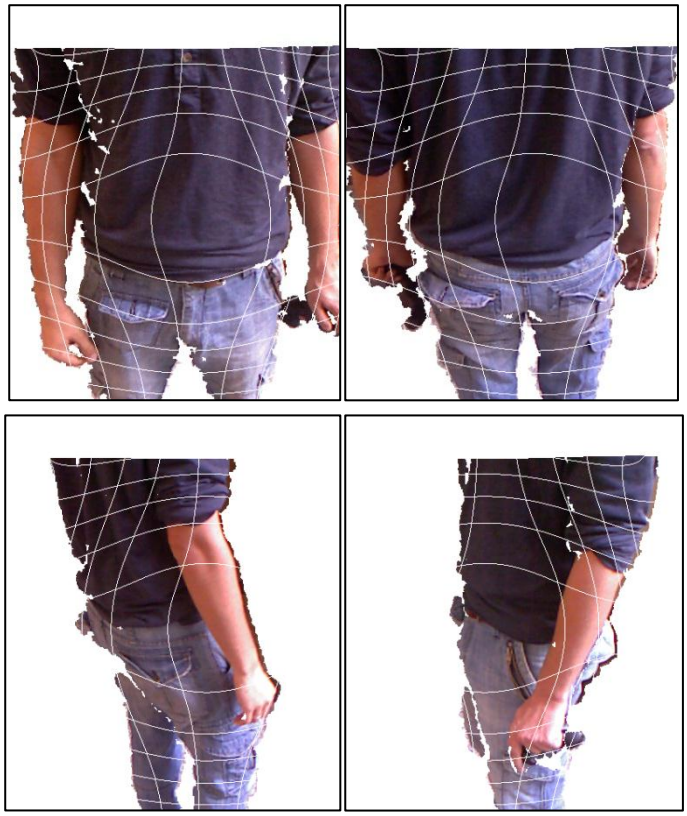

Abbildung 42: Beispiele der Initialisierungsscreenshots aus verschiedenen Perspektiven

8.1.7 Programmablaufphase (runPhase)

Mit Beendigung der Initialisierungsphase und der daraus resultierenden Erstellung der 20 Vorlagen, kann nun die runPhase gestartet werden.

Die runPhase dient von nun an als Kernfunktion, von der aus alle weiteren Funktionen nacheinander gestartet werden. Zunächst wird wieder eine Filterung der Punktwolke durchgeführt, was bereits im vorherigen Abschnitt erläutert wurde (Kapitel 8.1.4). Im weiteren Verlauf wird das 3D-Punktwolkenbild in ein 2D-Bild umgewandelt (Kapitel 8.1.8). Dieser Schritt wird benötigt um die benötigten OpenCv-Algorithmen auf dieses neue Bild anwenden zu können. Ausgehend vom FastTemplateMatcher (Kapitel 3.2.7), wird nach einer erfolgreichen Detektion des Ziels der mittlere Abstand zu diesem berechnet. Diese Ergebnisse benutzt die robotControlAndPublish-Funktion nun, um den Roboter in die entsprechende Richtung zu bewegen. Zusätzlich wird diese Bewegung durch den Status der Sicherheitstaste des Controllers abgesichert, welcher in der safetyButton-Funktion überprüft wird.

8.1.8 Umwandlung von 3D- zu 2D-Bildern (convert3Dto2D)

Nachdem die Punktwolke erfolgreich gefiltert wurde, können nun die Abstandsberechnungen zur Person durchgeführt werden. Zu Beginn der Distanzberechnung wird eine 2D-Matrix mit einer Größe von 640x480 Punkten mit den RGB-Farbinformationen der Farbe Weiß (255; 255; 255) gefüllt. Eine reine weiße Farbe kommt so gut wie niemals vor, weswegen das später erstellt Farbhistogramm diese Informationen ausblenden kann und somit nur die wirkliche Person erfasst wird.
Eine gesamte ungefilterte RGB-Punktwolke besteht aus XYZ-Punkten, welche bestimmte Farbinformationen haben. Zusammengefasst ergeben sie somit ein gesamtes Bild. Diese 3D-Punktwolke umgewandelt in ein 2D-Bild kommt einem Screenshot gleich. Bei einer gefilterten 3D-Punktwolke befinden sich die Punkte allerdings nicht mehr eindeutig an ihren Positionen. Diese müssen zunächst mittels einer definierten Formel in 2D-Punkte umgerechnet werden (Formel 6).

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Formel 6: Basisformel zur Umwandlung eines 3D-Punktes in einen 2D-Punkt (nach OpenCV, 2011)

Wenn XYZ die Koordinaten eines 3D-Punktes im Weltkoordinatensystem sind, dann entsprechen u und v den projizierten 2D-Punkten als Pixel dargestellt. Die A-Matrix ist dabei die Kameramatrix oder auch Matrix der intrinsischen Parameter genannt. Die Koordinaten (cx, cy) geben üblicherweise den Bildmittelpunkt an und die Parameter (fx, fy) entsprechen dabei der fokalen Länge der Kamera. Die achsenspezifische Rotations-, Translationsmatrix beschreibt die Veränderung der Kamera zur erfassten statischen Szene oder andersherum, die Veränderung der Szene in Relation zur statischen Kamera. Reale Kameras haben meistens eine gewisse radiale $(k1, k2, k3)$ und eine geringe tangentiale $(p1, p2)$ Verzerrung, welche in die Berechnung mit aufgenommen werden sollte. Somit ergibt sich die nachfolgende Formel 7 zur Umrechnung eines 3D-Punktes in 2D-Koordinaten.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t\ ; \qquad x' = \frac{x}{z}; \qquad y' = \frac{y}{z}\ ;$$

$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2\,p_1 x' y' + p_2(r^2 + 2x'^2)$$

$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2\,p_1 x' y' + p_2(r^2 + 2y'^2)$$

$$where\ \ r^2 = x'^2 + y'^2$$

$$u = f_x * x'' + c_x$$

$$v = f_y * y'' + c_y$$

Formel 7: Erweiterte Formel zur Umwandlung eines 3D-Punktes in einen 2D-Punkt (nach OpenCv, 2011)

Dabei wird zunächst einmal die x- und y- Koordinaten durch die z-Koordinate geteilt um beide mit den gleichen Voraussetzungen weiter bearbeiten zu können. Anschließend werden die radialen und tangentialen Verzerrungsparameter entsprechend aufsummiert und mit den Koordinaten jeweils multipliziert. Dabei entspricht Parameter r der Hypotenuse in einem Dreieck und somit der Distanz des Punktes. Schließlich werden die Koeffizienten für den Bildmittelpunkt zum Ergebnis hinzuaddiert, bzw. die Koeffizienten für die fokale Länge hinzumultipliziert. Das abschließende Ergebnis entspricht nun einer Umwandlung eines 3D-Punktes in ein 2D-Punkt.
Um nun wieder ein Bild zu erhalten, müssen alle Werte in Integer-Werte umgewandelt werden, da sie vorher, aufgrund der gegebenen Kameraparameter, in float-Werten berechnet wurden. Des Weiteren muss im Vorfeld jeder Farbpunkt in seine RGB-Farben aufgespalten werden, um die richtigen Farbinformationen berechnen zu können, da die Point Cloud Library, RGB-Werte in einer 32-Bit float Variable speichert. Dies geschieht mittels Bit Shifting der Farbwerte in der unsigned-Integer-32bit-Variable.

```
uint32_t rgb_val_;
memcpy(&rgb_val_, &(pt.rgb), sizeof(float));
uint8_t r_ = (uint8_t)((rgb_val_ >> 16) & 0x000000ff); //ergibt den roten Farbwert
uint8_t g_ = (uint8_t)((rgb_val_ >> 8) & 0x000000ff); //ergibt den grünen Farbwert
uint8_t b_ = (uint8_t)((rgb_val_) & 0x000000ff); //ergibt den blauen Farbwert
```

Nun können alle 2D-Punkte mit den richtigen Farbinformationen in die vorher erstellte Matrix übertragen werden. Das Ergebnis dieser Berechnung ist in Abbildung 43 zu sehen.

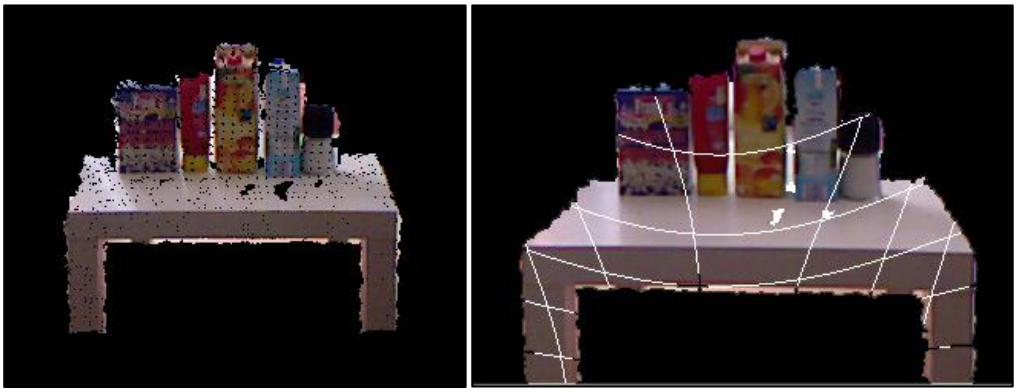

Abbildung 43:Umwandlung einer 3D-Punktwolke (links) in ein 2D-Bild (rechts)

Auffällig beim 2D-Bild sind Wellen, welche sich teilweise durch das Bild ziehen. Diese treten durch einen Sprung in der Berechnung auf, welcher daher kommt, dass integer-Werte immer aufgerundet werden. An manchen Stellen hinterlassen diese Sprünge eine undefinierte Stelle ohne Farbinformationen. Dies führt anschließend zu den Wellen. Diese beeinträchtigen allerdings nicht den weiteren Verlauf der Bildbearbeitung da dieser Fehler nur minimal ist und das Gesamtergebnis nicht stört.
Dieses 2D-Bild kann nun an die weiteren OpenCv Funktionen weitergegeben werden. Hierbei werden im weiteren Verlauf bestimmte Veränderungen an dem Bild vorgenommen um ein leichteres Tracking zu ermöglichen.

8.1.9 Fast Template Matching (fastTemplateMatcher)

Wie bereits in Kapitel 3.2.7 erläutert, verwendet der FastTemplateMatcher eine Vorlage eines Objektes, um dieses in einem aktuellen Bild wiederzufinden. Außerdem wurde dieser mithilfe der InitiScreenshot-Funktion dahingehend erweitert, dass er eine Person aus 20 Vorlagenbildern heraus erkennen kann. Sollte er die Zielperson in einem Bild nicht mehr finden können, so verwendet er die nachfolgende Vorlage und vergleicht diese nun mit dem aktuellen Bild. Wird dabei eine Übereinstimmung gefunden, so wird die Freigabe zur weiteren Bearbeitung erteilt. Dabei Berechnet die calculateMeanDistance-Funktion nun den Abstand zur Zielperson und gleichzeitig werden auch die genauen Koordinaten im Bild ermittelt. Diese Informationen, zusammen mit der Freigabe des FastTemplateMatchers, werden nun der Robotersteuerung zur Kalkulation der richtigen Bewegungsbefehle übergeben.

8.1.10 Distanzberechnung (calculateMeanDistance)

Um die Distanz zur Person berechnen zu können, wird nun die gefilterte Punktwolke betrachtet. Jeder Punkt im Bild hat bestimmte Werte für X, Y und Z. Aus der Summe aller sichtbaren Punkte wird ein Mittelwert berechnet. . Somit ergibt sich ein ausreichend guter Wert, welcher für die Distanz zur Person vollkommen ausreichend ist.

8.1.11 Robotersteuerung (robotControlAndPublish)

Im Kapitel der Robotersteuerung wurden bereits die Möglichkeiten erläutert dem Roboter, oder alternativ einer Simulationsumgebung, Bewegungsbefehle zu schicken. In diesem Kapitel sollen nun die speziellen Bewegungsmuster, basierend auf den Positionsberechnungen, aus dem FastTemplateMatcher, erklärt werden.
Um dem Roboter keine ungültigen Informationen zu übermitteln, wurden diese entsprechend eingeschränkt. Dabei wurden in X-Richtung anhand von Experimenten, folgende Regeln festgelegt:

- Ist das Ziel näher als 76 cm => fahr mit 0,2 rad/s zurück
  `cmd_vel.linear.x = -0.2;`
- Ist das Ziel weiter als 84 cm => fahr mit 0,2 rad/s vorwärts
  `cmd_vel.linear.x = +0.2;`
- Hat das Ziel einen Abstand von 77-83 cm => bleib an deiner aktuellen Position
  `cmd_vel.linear.x = 0.0;`

In theta-Richtung wurden entsprechend ähnliche Regeln definiert. Diese Regeln beziehen sich immer auf eine Veränderung des Mittelpunktes des Objektes in Bezug zur relativen Bildmitte:

- Weicht das Ziel mehr als 3 cm von der Bildmitte ab => dreh nach rechts mit 0,3 rad/s
  `cmd_vel.angular.z = +0.3;`
- Weicht das Ziel weniger als 3cm von der Bildmitte ab => dreh nach links mit 0,3 rad/s
  `cmd_vel.angular.z = -0.3;`
- Befindet sich das Ziel in einer Umgebung von -3 bis +3 cm relativ zur Bildmitte gesehen => bleib an deiner aktuellen Position stehen
  `cmd_vel.angular.z = 0.0;`

In beiden Fällen wird nach der Entscheidung, welche Regel ausgeführt wird, die Nachricht an den Roboter übermittelt. Da die Nachrichten direkt vom Roboter verarbeitet werden, kann es zu ruckelnden nicht flüssigen Bewegungen kommen. Eine Funktion, welche eine langsame Anfahrt und anschließend eine kontinuierliche Weiterfahrt ermöglicht, schafft hierbei Abhilfe.

8.1.12 Sicherheitsschalter (safetyButton)

Durch die Verwendung des Controllers können vor allem der Roboter selber und das Programm in einem definierten Maße gesteuert werden. In diesem Falle wurden zwei Knöpfe mit Zusatzfunktionen belegt, die es einmal ermöglichen, die automatische Bewegung des Roboters einzuschränken und im anderen Falle die Initialisierungsphase zu starten. Dazu wird im Vorfeld zum Joy-Topic subscribed und die erhaltenen Informationen entsprechend ausgewertet. Hierbei beinhaltet die empfangene Message die Information, welcher Button gerade gedrückt wurde. Dadurch ist es möglich, ausgewählten Buttons

entsprechende Funktionen zuzuweisen. In diesem Fall wurde der Button oben-vorne-links (Abbildung 44) als Sicherheitsknopf definiert. Ist dieser gedrückt, kann der Roboter automatisch fahren; wird er losgelassen, schaltet das System wieder auf manuelle Steuerung um. Der Button oben-vorne-rechts (Abbildung 44) dient dabei zum Starten der initphase-Funktion.

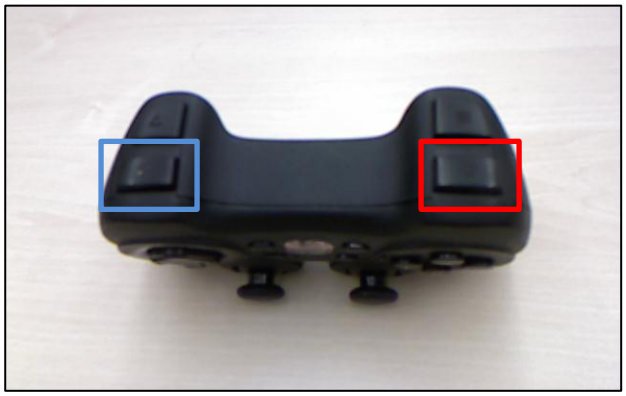

Abbildung 44: Sicherheits- (links, blauer Kasten) und Initialisierungsschalter (rechts, roter Kasten) des Controllers

8.2 Version 2 – Tiefenbild + Particle Filter

Noch während der Bearbeitung der Version 1 des Gesamtsystems, fielen die übermäßig hohe Rechenleistung bei der Bearbeitung der Punktwolke und der hohe Programmieraufwand auf. Schließlich wurde der Entschluss gefasst, eine alternative zu suchen und diese wurde schlussendlich auch in der Verwendung einer Kombination aus dem Tiefenbild der Kinect-Kamera und dem Particle Filter gefunden. Dieses Kapitel zeigt nun im weiteren Verlauf, alle benötigten Funktionen die zur Erfassung und Detektion der richtigen Person durchgeführt wurden.

Multithreading

Durch die Kombination von Farb- und Tiefenbild für die Verwendung in der Version 2 des Gesamtsystems wird eine Methodik benötigt mehrere Prozesse parallel zu verarbeiten. Die hier verwendete Methode ist das Multithreading. Dies wird genau dann der Fall sein, wenn z.B. ein Farb- und parallel dazu ein Tiefenbild von der Kamera übernommen und bearbeitet werden. Wird ein Bild, welches sich gerade in Bearbeitung befindet, erneuert, kann es zu einem Absturz des Programmes kommen, da die aktuelle Speicheradresse nicht mehr eindeutig zur Verfügung steht. Deswegen ist es nötig den Zugriff auf das Bild für den Zeitraum der Bearbeitung zu sperren. Dies geschieht mit den boost:mutex: Parametern lock und unlock, welche entsprechend vor und nach einer kritischen Stelle eingesetzt werden. Zusätzlich wird über die Verwendung der sogenannten Semaphoren eine Fortführung des Codes erst ermöglicht, wenn beide Threads abgeschlossen sind und bestimmte Variablen gesetzt wurden. Semaphoren geben die Programmbearbeitung erst

dann frei, wenn diese auf 1 gesetzt wurden. Dies geschieht z.B. dann, wenn ein Farbbild erfolgreich erstellt wurde und dem System nun zur Verfügung steht.

8.2.1 Ablauf des Programms im System

Das Programm besitzt verschiedene Stufen der Ausführung. Im Nachfolgenden werden alle Programmabschnitte und ihre Funktionen im Detail erklärt. Die grundlegenden Funktionen wurden bereits in den Kapiteln 3.2 - Betrachtete Methoden, 4.2.3 - Tiefenbild Filterung, 5 - Particle Filter und 7 - Robotersteuerung eingehend erklärt und werden hierbei nur noch der Vollständigkeit halber erläutert. Das Ablaufdiagramm für die Version 2 des Gesamtsystems ist in Abbildung 45 zu sehen.

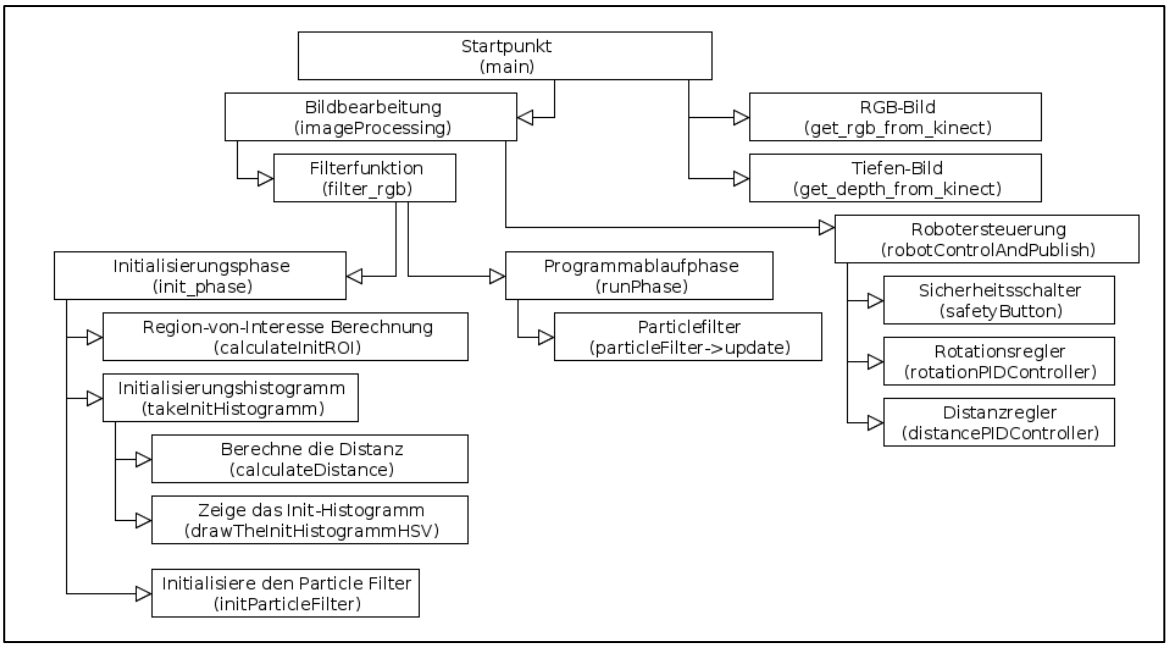

Abbildung 45: Ablaufdiagramm der zweiten Version des Hauptprogramms

8.2.2 Startpunkt (main)

In der Main-Funktion werden die entsprechenden Nodes und Subscriber erstellt, welche sich zu den Publishertopics der Farb- und Tiefenbilder der Kamera verbinden. Hierbei handelt es sich um eine Erstellung eines Multithreads, da beide Kamerainformationen (Farbe und Tiefe) parallel abgerufen und bearbeitet werden müssen. Dies ist wichtig um auch die richtigen Farbinformationen den korrespondierenden Tiefeninformationen übergeben zu können.

8.2.3 RGB- und Tiefenbild (get_rgb_from_kinect- und get_depth_from_kinect)

Aus der Main-Funktion heraus werden die Funktionen get_rgb_from_kinect und get_depth_from_kinect aufgerufen. Diese beiden Funktionen bereiten die erhaltenen Farb- und Tiefenbilder der Kinect für die Weiterbearbeitung durch OpenCv vor. Dazu wird die sog. CvBridge des Ros-Systems verwendet. Diese wandelt Ros-Messages, in diesem Fall handelt es sich um eine Message mit Bildinformationen, in ein für OpenCv konformes Format um. Somit können die erhaltenen Bildinformationen direkt in eine Bild-Matrix eingelesen werden und stehen dem Programm von nun an als korrektes Bild zur Verfügung.
Um die Synchronisation zwischen den einzelnen Bild-Threads zu ermöglichen, werden die sogenannten Semaphoren eingesetzt, um einen weiteren Programmablauf erst zu ermöglichen, wenn ein korrektes Bild erhalten wurde. Dies hat den Hintergrund, dass kein Bild, welches sich momentan in Bearbeitung odereventuell in Manipulation befindet, durch ein neues Bild ersetzt wird. Eine solche Speicherüberschreibung kann einen Programmabsturz zur Folge haben. Eine weitere Sicherheit ist die simple Abfrage durch boolesche Variablen newRGBImg und newDeapthImg, ob ein Bild erhalten wurde. Sind beide Bilder, Farbe und Tiefe korrekt übermittelt und dem Speicherbereich zugewiesen worden, werden beide Variablen auf „true“ gesetzt und der Programmablauf kann fortgeführt werden. Die verschiedenen Ansichten sind in Abbildung 46 zu sehen, wobei links das Tiefenbild und rechts das RGB-Bild dargestellt sind.

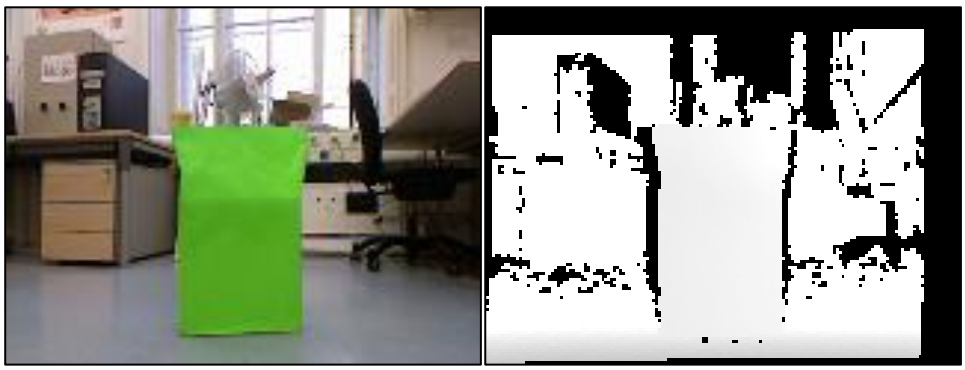

Abbildung 46: Ausgabe der reinen Bilddaten, links, RGB-Bild; rechts, Tiefenbild

8.2.4 Bildbearbeitung (imageProcessing)

Die ImageProcessing-Funktion dient der Verwaltung der Haupt-Funktionen: initPhase, runPhase und robotControlAndPublish. In dieser Funktion werden zunächst wieder die möglichen Speicherüberschreibungen so lange gelockt, wie das Programm mit der Bearbeitung der beiden Bilder beschäftigt ist. Zusätzlich zur Sicherheit der Roboterbewegungen kann die initPhase auch mithilfe des Controllers gestartet werden. Dies hat den Vorteil, dass sich eine Person direkt vor die Kamera stellen und erfassen

lassen kann, ohne dass andere Personen den Programmablauf über die Tastatur steuern müssen.
Wurde die initPhase-Funktion erfolgreich abgearbeitet, wird die runPhase-Funktion und robotControlAndPublish-Funktion durchgeführt und die intiPhase-Funktion wiederum gesperrt. Es besteht allerdings nochmals die Möglichkeit durch Verwendung des Controllers oder der Tastatur, die initphase-Funktion erneut aufzurufen. Dadurch können mehrere Objekte getestet werden, ohne das Programm erneut starten zu müssen.
Im Abschluss dieser Funktion werden die Variablen newRGBImg und newDepthImg wieder auf „false“ gesetzt und die erneute Bildaufnahme unlockt.

8.2.5 Sicherheitsschalter (safetyButton)

Die Verwendung des Controllers ist auch in dieser Version des Gesamtsystems fester Bestandteil des gesamten Programmes. Die Erklärung dazu ist nahezu äquivalent mit der aus Kapitel 8.1.12, bis auf den einzigen Unterschied, dass hierbei der Button oben-vorne-rechts (Abbildung 44) zum Starten der initphase-Funktion und somit zur Erstellung des Initialisierungshistogramms und des Initialisierungsmodell des Particle-Filters verwendet wird.

8.2.6 Initialisierungsphase (init_phase)

Neben der Erstellung des Initialisierungshistogramms wird in der initPhase auch das Initialisierungsmodell für den Particle-Filter erstellt und diesem anschließend übergeben. Diesen wichtigen Funktionen voran wird das aktuelle Tiefen- und Farbbild nach einem definierten Abstand gefiltert. Solange die Initialisierungsphase läuft, wird ein Histogramm des aktuellen Objektes erstellt. Ist dieses Objekt hinreichend genau erfasst, kann die Initialisierungsphase beendet werden und das nun erhaltene Initialisierungshistogramm wird mit weiteren Parametern dem Particle-Filter übergeben.

8.2.7 Filterfunktion (filter_rgb)

In der filter-Funktion werden die Tiefen und Farbbilder nach einem definierten Abstand von 1.5m gefiltert. Somit wird nur der Bereich, welcher von Interesse ist, für die späteren Funktionen zur Verfügung gestellt und der Hintergrund gelöscht. Dazu wird jeder Punkt des Tiefenbildes auf seinen Abstand hin untersucht. Befindet er sich noch im definierten Bereich, werden die Koordinaten dieses Punktes dem Farbbild übergeben und in eine neue Farbbildmatrix gefüllt. Die Anzahl dieser Punkte wird parallel dazu mitgezählt und dient im Anschluss der takeInitHistogramm-Funktion zur Erstellung des Initialisierungshistogramms. Punkte, welche sich außerhalb des definierten Bereiches befinden, werden im Tiefenbild auf 0 gesetzt und bekommen im Farbbild die Farbinformation eines schwarzen Punktes. Die Ergebnisse der Filterung sind in Abbildung 47 zu erkennen. Dabei wurden links das gefilterte RGB-Bild und rechts das gefilterte Tiefenbild dargestellt.

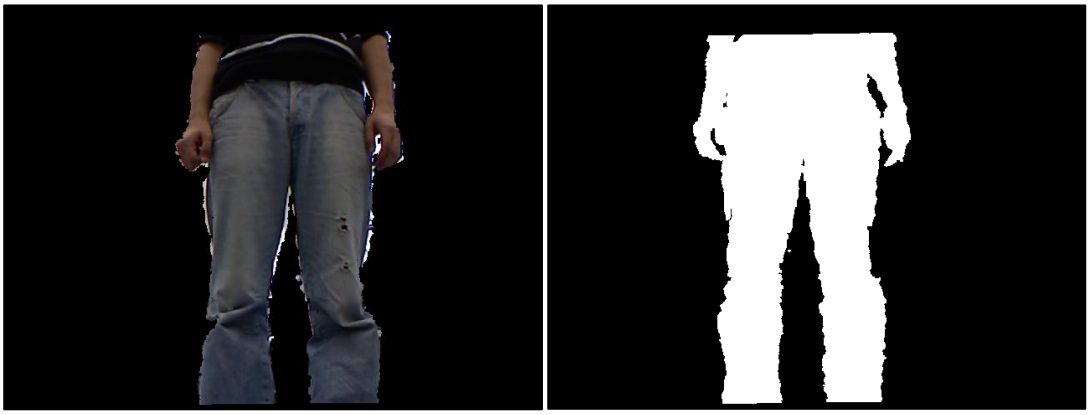

Abbildung 47: Gefilterte Bilder der Kamera, RGB-Bild (links) und Tiefenbild (rechts)

8.2.8 Region-von-Interesse Berechnung (calculateInitROI)

Die Funktion calculateInitROI gibt in diesem speziellen Fall einen Region-of-Interest zurück, in welchem sich das gewünschte Objekt befindet. Dazu wird das auf 1.5 m gefilterte Tiefenbild verwendet, um den Suchbereich einzuschränken und das Ergebnis möglichst sauber zu erhalten. Die OpenCv-Funktion FloodFill übernimmt hierbei die Aufgabe, das Objekt entsprechend zu erfassen und dessen Größe einzuspeichern. Die Größe des Objekts wird nun durch ein Rechteck repräsentiert. Dieses hat außerdem noch genau definierte Koordinaten im Bild und dient der takeInitHistogramm-Funktion zur Einschränkung des Suchbereichs nach dem Objekt.

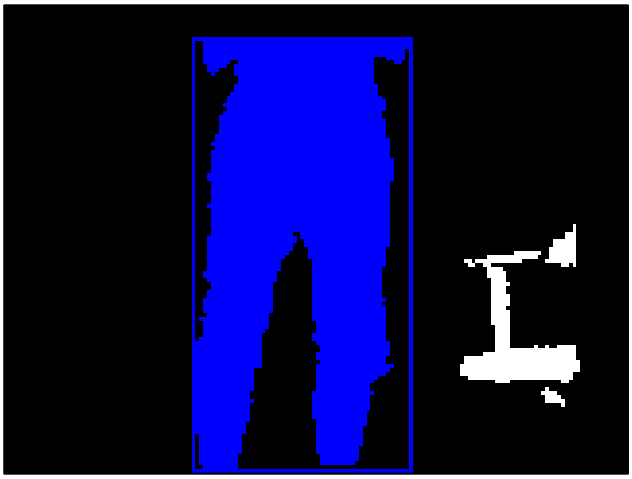

Abbildung 48: Erfasste Person mit der Floodfill-Funktion

8.2.9 Initialisierungshistogramm (takeInitHistogramm)

Eine der wichtigsten Funktionen des Programms ist die der Erstellung des Initialisierungshistogramms. Dabei wird das momentane Farb- und Tiefenbild der Kamera auf den Bereich getrimmt, welcher durch die calculateROI-Funktion definiert wurde. Somit erhalten wir ein Region-of-Interest Farb- und Tiefenbild. Dieses hat eine bestimmte definierte Größe und beinhaltet das gewünschte Objekt. Solange sich das Objekt in einem

Abstand näher als 1.5 m befindet, werden dessen Farbinformationen in eine neue Matrix gefüllt, welche die Höhe 1 und die Breite n hat. Diese Matrix ist nicht darstellbar wie andere Bilder, dient allerdings der Histogrammfunktion zur Erstellung des richtigen Histogramms, da für diese die Größe und Breite einer Matrix nicht von Bedeutung sind. Parallel dazu wird ein Mittelwert aller Distanzen im ROI berechnet. Dieser dient im weiteren Verlauf dazu, dem Particle-Filter eine Initialisierungsdistanz übergeben zu können.
Eine weitere OpenCv-Funktion wandelt die erstellte Histogramm-Matrix zunächst in ein HSV-Format um, woraus anschließend das Histogramm berechnet wird. Dieser Schritt ist nötig, da dass HSV-Farbmodell eine geringere Beeinflussung gegenüber Helligkeitsunterschiede aufweist, als es beim RGB-Farbmodell der Fall ist.

8.2.10 Programmablaufphase (run_phase)

Nach der erfolgreichen Erstellung des Initialisierungshistogramms wird die runPhase-Funktion gestartet. In ihr wird der Particle Filter ausgeführt, welcher für den weiteren Ablauf des Programmes benötig wird.

8.2.11 Particle Filter (particleFilter)

Wie bereits in Kapitel 5 erläutert, wird der Particle Filter effektiv eingesetzt um die Person im Bild zu verfolgen. In der Initialisierungsphase werden ihm unter anderem das Initialisierungshistogramm, die Initialisierungs-ROI und die Initialisierungsdistanz übergeben. Dadurch kann er flexibler, schneller und stabiler eine Person verfolgen als es bei der Version 1, der FastTemplateMatcher konnte.

initParticleFilter

In diesem Fall wird dem Modell im Vorfeld ein Initialisierungs-Histogramm, eine Initialisierungs-Region-of-Interest Größe und eine Initialisierungsdistanz übergeben. Des Weiteren wird die Startkoordinate auf den Mittelpunkt des Bildes gesetzt, da davon ausgegangen wird, dass sich das Objekt in dessen Zentrum befindet.
Mithilfe der FloodFill-Funktion (Kapitel 3.2.11) kann nun ein Objekt, welches von Interesse ist, erfasst werden (Abbildung 48). Durch diese ist nun auch die Größe, Breite und die Position des Objektes bekannt.

Observation

Zusätzlich zur eigentlichen Observation Funktion des Particle Filters, wird auch eine graphische Darstellung des aktuellen Histogramms implementiert (Abbildung 49). Diese dient dazu, die Ergebnisse zu kontrollieren. Dazu wird das aktuelle Bild jeweils in 3 Bereiche gesplittet, sodass jeder Farbkanal (HSV) ein eigenes Histogramm erhält.

Abbildung 49: Ergebnisse der Histogramme, währen der Initialisierungs- und der Runphase

calculateMoments

Aus den gesamten Berechnungen eines Durchlaufes wird nun die bestmögliche Position für eine erneute Kalkulation ermittelt. Dazu wird die Mittlere Position aus allen Particles berechnet und dient somit dem Roboter als Hinweis darauf, wo sich die Person nun befindet.

visualizeParticles

In der visualizeParticles-Funktion werden die einzelnen Particles graphisch dargestellt. Dies dient in erster Linie dazu, den richtigen Programmablauf zu gewährleisten. Außerdem gibt es direkt Rückschlüsse darüber, welches Objekt aktuell vom Particle Filter erfasst wird (siehe Abbildung 31).

8.2.12 Robotersteuerung (robotControlAndPublish)

Die robotControlAndPublish-Funktion dient zur Steuerung des Roboters. In ihr werden alle erhaltenen Distanz- und Rotationsinformationen bearbeitet und dem Roboter übergeben. Der implementierte PID-Regler steuert dabei die Berechnungen dieser beiden Informationen. Ausgehend von den Berechnungen des Reglers werden dem Roboter verschiedene Steuerungsbefehle geschickt, welche ihn vorwärts, rückwärts, nach links, nach rechts bewegen oder ihn auf der Stelle stehen bleiben lassen können. Um dies zu erreichen, werden wie beim Controller die Geschwindigkeitsparameter über das CmdVel-Topic an den Roboter verschickt. Beide Parameter erhalten ihre Geschwindigkeitsbefehle in Radiant pro Sekunde (rad/s). Dabei dient cmd_vel.linear.x zur Steuerung der Linearbewegungen und cmd_vel.angular.z zur Steuerung der Rotationsbewegung.

8.2.13 Abstands- und Rotationsregler (distancePIDController- und rotationPIDController)

Zur Steuerung des Roboters müssen die einzelnen Distanzwerte zunächst geprüft und einem bestimmten Befehl zugeordnet werden. Aus diesem Grund wird ein sogenannter PID-Regler eingesetzt, welcher ideal für diese Aufgabe ist. Dabei handelt es sich um eine Kombination aus einem Proportional-, Integral- und Derivativen-Regler. Die beiden Funktionen zur Steuerung des Roboters über den PID-Regler werden nach folgendem einfachen Pseudocode erstellt und ausgeführt. Dabei entspricht der Parameter Kp dem proportionalen Anteil des Reglers. Äquivalent dazu stehen die Parameter Ki und Kd, jeweils für den integralen und derivativen Anteil des Reglers.

1. vorheriger_fehler = gesetzter_wert (z.B. 1 m) – aktuelle_position (z.B. 1,5 m)
2. intergral = 0
3. startpunkt
4. fehler = gesetzter_wert – aktuelle_position
5. integral = integral + (fehler * dt)
6. derivative = (fehler – vorheriger_fehler)/dt
7. ergebnis = (Kp * fehler) + (Ki * integral) + (Kd * derivative)
8. vorheriger_fehler = fehler
9. warte(dt)
10. gehe zum startpunkt

Der Unterschied in den beiden Funktionen ist lediglich die Einstellungen der Parameter und die Werte, welche ihnen im Vorfeld übergeben werden. Die distancePIDController-Funktion erhält ihre Distanzwerte aus der Mittelwertberechnung der calculateDepthError-Funktion des Particle-Filters. Äquivalent dazu wird die aktuelle Position der rotationPIDController-Funktion aus dem Mittelwert der x-Koordinaten aller gültigen Particles ermittelt.
Da die Werte für die Parameter Kp, Ki und Kd im Vorfeld definiert werden müssen, wurden verschiedene Test mit einer Reihe von werten durchgeführt. Dabei wurden verschiedene Kombinationen ausprobiert und die bestmögliche Kombination in das Gesamtsystem integriert. Bei beiden PID-Reglern gilt dasselbe Schema, wie die Ergebnisse verarbeitet werden. Dabei gibt es grob gesehen drei Stadien, welche entsprechend anders interpretiert werden können. Ein Wert kleiner 0 bedeutet in diesem Fall, das Objekt befindet sich zu weit weg vom Roboter oder in der rechten Bildhälfte. Alternativ dazu bedeutet ein Wert größer 0, dass das Objekt sich zu nah am Roboter oder sich in der linken Bildhälfte befindet. Bei einem Ergebnis von 0 befindet sich das Objekt im definierten Abstand zum Roboter und in der Bildmitte. Da der Wert 0 durch die relativ groben Bewegungen des Roboters kaum erreicht wird, wurde ein Threshold eingebaut, um ein gewisses Spiel bei der Erreichung der optimalen Position zu ermöglichen.

8.3 Ablauf des Programms für den User

Der Aufwand für den User sollte so gering wie möglich gehalten werden. Weswegen er im Idealfall nur vor dem Roboter kurz stehen sollte, um dann direkt einkaufen gehen zu können.

8.3.1 Person stellt sich vor die Kamera

Damit das System eine Person richtig erfassen kann, sollte diese sich direkt vor der Kamera in einem definierten und festgelegten Abstand befinden. Aufgrund der Bauweise der Kinect darf der Abstand zu dieser nicht näher als 0,5m betragen. Erst ab diesen kann die 3D-Position ermittelt werden. Um die Rechenleistung möglichst gering zu halten wird der sichtbare Bereich der Kamera auf einen Abstand in z-Richtung von 1,5m eingeschränkt. Um den Sichtbereich weiter einzuschränken, werden alle Informationen, welche sich von Mittelpunkt der Kamera gesehen, in +X- und -X-Richtung in jeweils 0,5m Abstand befinden, herausgefiltert. Lediglich in Y-Richtung wird das gesamte Bild verwendet, da diese Informationen wichtig für die Auswertung sind.
Ausgehend von der Version 1 des Programms, muss die Person in den ersten Sekunden der Initialisierungsphase ein 360°-Drehung durchführen. Diese wird, wie bereits erwähnt (siehe Kapitel 8.1.6) benötigt, um die dem FastTemplateMatcher genügend Vorlagen aus allen Perspektiven liefern zu können. Die Version 2 des Programms benötigt dies nicht. Die Person sollte sich nur möglichst aufrecht im definierten Bereich aufhalten, damit das Initialisierungshistogramm erstellt werden kann.

8.3.2 Person steuert den Roboter mittels Controller

In der Testphase des Projektes, kann der Roboter zusätzlich mit dem Controller gesteuert und die Erstellung des Initialisierungshistogramms darüber gestartet werden. So kann die Person den Controller bequem in der Hand halten und es muss keine zweite Person die Tastatur des Laptops bedienen.
Wurde die Initialisierungsphase abgeschlossen, so kann die automatische Personenverfolgung und somit die Bewegung des Roboters nur gestartet werden, wenn der entsprechende Sicherheitsknopf am Controller gedrückt gehalten wird. Ein Loslassen dieses, unterbindet die automatische Bewegung und schaltet auf die manuelle Steuerung um. Damit ist immer gewährleistet, dass die Roboterbewegungen im Falle eines Problems stets kontrollierbar bleiben.

8.3.3 Person bewegt sich

Aufgrund der Bauweise und der Geschwindigkeitsbeschränkungen des Robotertreibers ist es ihm nicht möglich, hohe Geschwindigkeiten über einen größeren Zeitraum einzuhalten. Dieser Umstand führt dazu, dass die Person dazu angehalten ist sich möglichst langsam fortzubewegen, damit der Roboter eine Möglichkeit hat ihm zu folgen.

8.3.4 Person beendet das Programm

Will die Zielperson nun die Bewegungsverfolgung wieder beendet, muss sie lediglich den Sicherheitsknopf des Controllers freigeben. Nachdem der Roboter zum Stillstand gekommen ist, kann das Programm beendet werden. Dies geschieht in der Prototypenphase in der der Entwicklungsumgebung Eclipse. Im fertigen System hingegen, würde der Roboter die Bewegungsverfolgung über einen Knopfdruck z.B. am Roboter beendet werden. Dieser würde nun automatisch an seine definierte Position zurückfahre.

8.4 Versuche und Evaluierungen der beiden Systeme

Im weiteren Verlauf werden die durchgeführten Versuche und die darauffolgende Evaluation der Ergebnisse veranschaulicht. Kapitel 8.1 befasst sich dabei mit dem Versuchsaufbau der Version 1 des Gesamtsystems, wohingegen Kapitel 8.2 sich mit dem ausführlicheren Versuchsaufbau der Version 2 des Gesamtsystems befasst.

8.4.1 Version 1 des Gesamtsystems

Aus den verschiedenen durchgeführten Tests mit der Version 1 des Gesamtsystems konnte relativ schnell die erfolgreiche Umsetzung des Grundprinzips erkannt werden. So folgte der Roboter einer Person, wenn auch nicht ganz eindeutig und nach wie vor träge durch die Testumgebung. Durch die Erweiterung des FastTemplateMatcher mit 20 Initialisierungsscreenshots von allen Perspektiven, wurde eine noch schneller Erfassung und Verfolgung der Zielperson ermöglicht.

Durch die Implementierung der 3D-Punktwolke in Kombination mit dem FastTemplateMatcher begann das System allerdings immer langsamer zu werden. Vor allem in Momenten, in denen der FastTemplateMatcher keine Übereinstimmung mit seiner Vorlage mehr erkennen konnte, verlangsamte sich das Gesamtsystem auf 1-2 Bilder/s. In der Zeit bewegte sich die Zielperson allerdings weiter und somit auch außerhalb des möglichen Suchbereiches. Dadurch wurde es nötig, sich möglich langsam vor dem Roboter zu bewegen, sodass dieser einer Zielperson auch effektiv auch folgen konnte. Wurde eine Zielperson hingegen einmal erfolgreich erkannt und bewegte sie sich entsprechend langsam vor dem Roboter, so konnte eine relativ gute Verfolgung dieser durchgeführt werden.
Aus diesen Tests wurde der Schluss gezogen, dass die Version 1 des Gesamtsystems nicht schnell genug ist um einer Person effektiv zu folgen. Die langsame Kommunikation mit dem Roboter und die daraus resultierende zeitverzögerte Umsetzung der Bewegungsbefehle, verstärkten diese Problematik nochmal zusätzlich.
Schließlich wurden keine weiteren Tests mehr mit der Version 1 durchgeführt, da nun an einer Verbesserung des Geschwindigkeitsproblems gearbeitet wurde. Als Resultat wurde schlussendlich die Version 2 des Gesamtsystems entwickelt.

8.4.2 Version 2 des Gesamtsystems

Die Version 2 des Gesamtsystems basiert wie bereits erläutert aus einer Kombination des Tiefenbildes der Kinect Kamera und dem Particle Filter. Dabei werden die Farbinformationen der Zielperson verwendet um diese möglichst eindeutig zu detektieren und zu verfolgen. Die Testumgebung war dabei ähnlich dem Testaufbau in Version 1 des Gesamtsystems. Hierbei wurden allerdings mehr Testdurchläufe mit verschiedenen Personen und Kleidungsstücken durchgeführt.

Zunächst wurde eine Person mit einer langen blauen Jeanshose erfasst und im Bild gesucht. Diese konnte mit einer 80-90% Wahrscheinlichkeit wieder im aktuellen Bild gefunden werden, solange sich keine anderen Objekte im Blickfeld der Kamera befunden haben. Der Roboter konnte entsprechend seiner Programmierung den Bewegungen der Zielperson relativ gut folgen. Das Problem der zeitverzögerten Kommunikation und der daraus resultierenden trägen Bewegungen blieb hierbei allerdings, wie bereits in Version 1 des Gesamtsystems, bestehen. Durch den Einsatz des Particle Filters, wurde die Zielperson allerdings um ein vielfaches schneller und robuster gefunden, ohne das im Vorfeld eine 360°-Aufnahme dieser gemacht werden musste (Vergleiche FastTemplateMatcher 3.2.7). Auch die Bewegung durch die Testumgebung konnte relativ zügig durchgeführt werden.

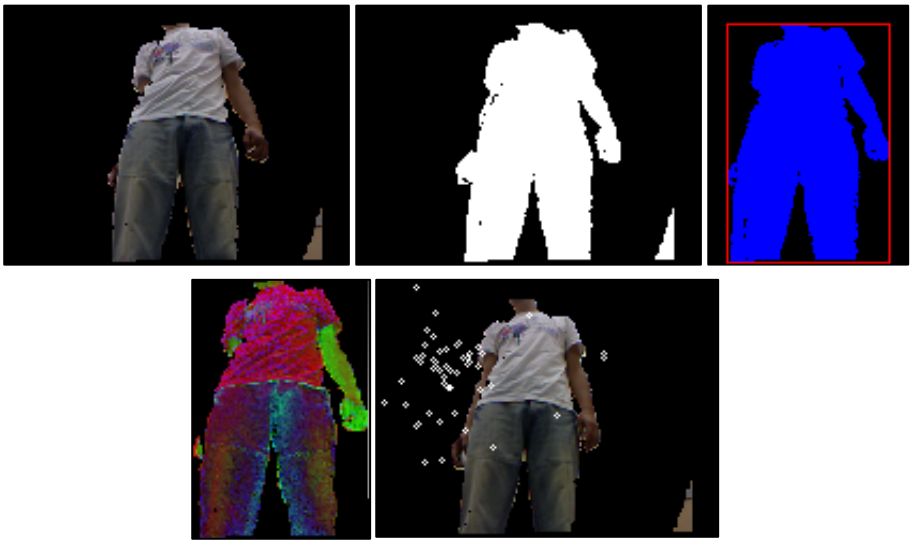

Abbildung 50: Testlauf der Version 2 des Gesamtsystems, einer Person mit einer langen blauen Jeanshose
oben-links: Initialisierungs-RGB-Bild; oben-mitte: Initialisierungs-Tiefenbild;
oben-rechts: Initialisierungs-Rechteck (roter Kasten) durch die FloodFill-Funktion
unten-links: aktuelles HSV-Bild; unten-rechts: erfasste Person durch den Particle Filter

Durch die Wahl der Farbdetektion als Basis für den Particle Filter ergaben sich allerdings im Laufe der Testphase einige Probleme. So besitzt die Kinect Kamera einen automatischen Weißabgleich, welcher dafür sorgt, dass die erfassten Bilder stets den natürlichen Farben entsprechen. Das diese Regelung allerdings auf mathematischen Kalkulationen basieren, kam es vor, dass die Kamera die tatsächlich vorkommenden Farben falsch interpretierte. Dies zeichnete sich vor allem durch stark bläuliche oder grünliche Bilder ab. Da die Verfolgung der Zielperson nun auf Farbhistogramme basiert, wurden die Berechnungen der aktuellen Histogramme durch diese Farbverschiebung komplett verfälscht. So kam es, dass eine 40% Übereinstimmung zum Initialisierungshistogramm in einem Schrank oder einem Karton errechnet wurde.

Um diesen Farbfehler weitgehend zu kompensieren, wurde zusätzlich die Distanz, als Gewichtungsfaktor für den Particle Filter eingebaut. Während der Initialisierungsphase befindet sich die Zielperson an einem definierten Punkt im Abstand von 1 m, welche nun der Initialisierungsdistanz des Particle Filters entspricht. Wird nun ein fälschlicherweise ein Schrank betrachtet und ergibt der Histogrammvergleich aufgrund der Farbverschiebung eine 40%ige Übereinstimmung, so wurde nun zusätzlich die Distanz zu dem Objekt berechnet. Befand sie der Schrank nun in einem Abstand, welcher stark von der Initialisierungsposition abwich, konnte eine falsche Gesamtberechnung vermieden werden.

Das Problem welches sich nun hierbei einstellte, war die Wahl der Gewichtung der einzelnen Funktionen. Bei einer Übereinstimmungsgewichtung von 50% zu 50% kam es vor, dass der Histogrammvergleich keinerlei Übereinstimmungen mit dem Initialisierungshistogramm errechnen konnte. Befand sich allerdings das fälschliche Objekt in einem Abstand, welcher genau der Initialisierungsposition entsprach, so wurde hierbei eine sehr große Übereinstimmung errechnet. Insgesamt hielt der Particle Filter das fälschliche Objekt nun für die Zielperson und der Roboter erhielt falsche Bewegungsbefehle.
Wurde nun hingegen die Übereinstimmungsgewichtung auf 70% zu 30% gelegt, so kam es zu erneuten Problemen in der Berechnung. Der Histogrammvergleich errechnete fälschlicherweise einen sehr guten Übereinstimmungswert, wohingegen die Distanzberechnung keinerlei Übereinstimmung errechnen konnte. Daraus ergab sich eine insgesamt gute Gesamtübereinstimmung und der Particle Filter bzw. der Roboter folgte nun dem falschen Objekt.

Schließlich wurde die Version 2 des Gesamtsystems mit weiteren Versuchspersonen getestet. Dabei wurde ein Versuchslauf mit einer Person durchgeführt, welche eine kurze Jeanshose trug (Abbildung 51). Die Schwierigkeit die sich dabei zeigte, war die Ähnlichkeit der Farben der Beine mit denen der umgebenden Schränke. Obwohl der Particle Filter des

Öfteren falsche Gesamtübereinstimmungen errechnete, konnte die Person relativ gut innerhalb der Testumgebung detektiert werden.

Abbildung 51: Testlauf der Version 2 des Gesamtsystems einer Person mit einer kurzen blauen Jeans
oben-links: Initialisierungs-RGB-Bild; oben-mitte: Initialisierungs-Tiefenbild;
oben-rechts: Initialisierungs-Rechteck (roter Kasten) durch die FloodFill-Funktion
unten-links: aktuelles HSV-Bild; unten-rechts: erfasste Person durch den Particle Filter

Um den Testlauf noch weiter zu erschweren, befand sich eine weitere Person mit einer kurzen sandfarbenen Jeanshose im aktuellen Bildbereich (Abbildung 52). Nichtsdestotrotz, wurde nur die Zielperson vom Particle Filter erfasst. Selbst wenn diese komplett aus dem Bild verschwunden ist, wurde keine Gesamtübereinstimmung mit der zweiten Person gefunden. Kam die Zielperson wieder ins Bild, erfasste der Particle Filter diese wieder erfolgreich und der Roboter konnte ihr wieder folgen.

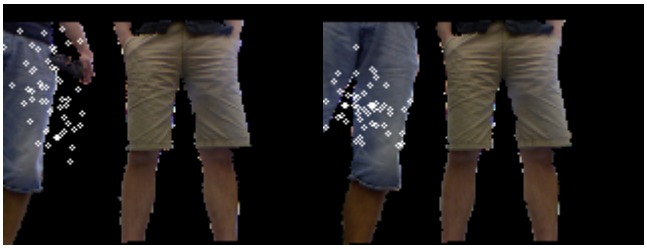

Abbildung 52: Korrekte Detektion der Zielperson mit kurzen blauen Jeans (links), trotz erschwerten Bedingungen der Testumstände durch eine weitere Person im Bild (rechts)

Eine weitere Versuchsperson war in diesem Fall eine Frau mit einem mittellangen farbigen Rock. Nach der Initialisierungsphase konnte der Roboter ihren Bewegungen unbeirrt folgen. Eine fälschliche Übereinstimmung mit einem Schrank oder einem anderen Objekt

konnte dabei nicht festgestellt werden. So wurde dieser Test erneut durch die Anwesenheit einer zweiten Person mit einer kurzen blauen Jeanshose erschwert. Dennoch blieb der Particle Filter die gesamte Zeit über auf der korrekten Zielperson und der Roboter folgte nur ihren Bewegungen.

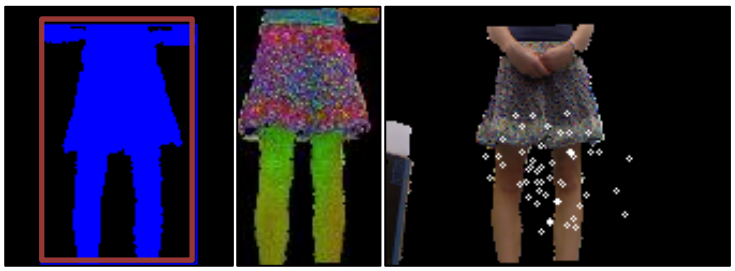

Abbildung 53: Testlauf der Version 2 des Gesamtsystems einer Person mit einem Rock
links: Initialisierungs-Rechteck (roter Kasten) durch die FloodFill-Funktion
mitte: aktuelles HSV-Bild; rechts: erfasste Person durch den Particle Filter

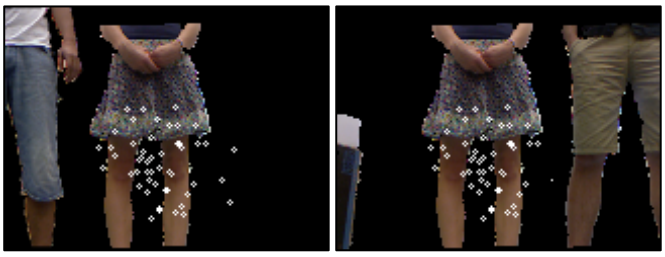

Abbildung 54: Korrekte Detektion der Zielperson mit Rock, trotz erschwerten Bedingungen der Testumstände durch eine weitere Person im Bild (links und rechts)

Wie diese Tests veranschaulichen konnten, bedarf es noch einiger Arbeit in diesem Bereich. Vor allem die Implementierungen von neuen Funktionen in den Particle Filter benötigen eine bessere Anpassung aufeinander. Aufgrund der kurzen Projektlaufzeit konnten keine weiteren Verbesserungen in diesem Bereich erzielt werden.

9 Zusammenfassung und Ausblick

Diese Masterarbeit befasste sich mit der Idee, einen mobilen Roboter als autonome Einkaufshilfe zu entwickeln. Dazu wurden in der anfänglichen Projektdefinition die einzelnen Projektziele und die anstehenden Herausforderungen fixiert und erörtert. Schließlich wurde ein Überblick über die nachfolgenden Kapitel dieser Arbeit gegeben. Das einführende Kapitel schloss mit einem kurzen Überblick über den aktuellen Stand der Technik und der Betrachtung zweier Projekte ab. Dabei diente in dem ersten Projekt ein selbstgebauter mobiler Roboter einer Person als Einkaufhilfe in einem Einkaufzentrum. Das andere Projekt befasste sich mit der Detektion und Verfolgung einer Person in einer Umgebung mit weiteren Personen und Hindernissen.

Schließlich folgte ein Überblick über die benutzten Basiskomponenten wie die verwendete Basis Software und Basis Hardware. Bei der Wahl der Basis Software wurde die Wahl der Komponenten vor allem auf Basis von Erfahrung (Wurde sie bereits erfolgreich in anderen Projekten eingesetzt?), Funktionalität (was kann alles mit ihr erreicht werden?), Standardisierung (welche wird Standardmäßig bei dieser Aufgabenstellung eingesetzt?) und Usability (wie benutzerfreundlich ist sie?) ausgewählt. So wird die quelloffene OpenSource Vision Bibliothek OpenCv als erfolgreiche Alternative zum kommerziell erhältlichen MatLab-Programm angesehen. Eine weitere quelloffene Software ist das Robot Operating System (ROS) und die Point Cloud Library (PCL). Mithilfe von ROS, kann nahezu jeder Roboter und mittlerweile eine große Auswahl von Hardwarekomponenten gesteuert, konfiguriert und benutzt werden. Durch die PCL ist es möglich, 3D-Punktwolken zu manipulieren und zu bearbeiten. Dies wird vor allem in der Version 1 des Gesamtsystems von großer Bedeutung sein. Programme können nun mit diese Softwarekomponenten mit der gleichen Entwicklungssprache C++ und in der gleichen Entwicklungsumgebung Eclipse programmiert und bearbeitet werden.

Bei der Wahl der Basis Hardware wurden auf abgewandelter Weise, die gleichen Kriterien wie bei der Wahl der Basis Software betrachtet. Hierbei spielte vor allem der Kostenfaktor eine große Rolle. So konnte auf einen mobilen Roboter zurückgegriffen werden, welcher sich bereits im Besitz des ACIN-Institutes der Technischen Universität Wiens befand. Dieser Pioneer-Roboter, konnte sich bereits in vorherigen Projekten erfolgreich bewähren und wird auch zu Übungszwecke in Unterrichtseinheiten eingesetzt. Der Einsatz von weiteren Sensoren zur Kollisionsvermeidung, -detektion und zur Navigation, welche am Roboter montiert sind, wurde im weiteren Verlauf dieser Masterarbeit geprüft. Neben dem Roboter, wurde auch die gewählte Kamera nach diesen Kriterien gewählt. Statt eines teuren Stereokameraaufbaus, wurde die neuartige Kinect Kamera verwendet. Diese wird eigentlich in Spielen für die XBOX360 der Firma Microsoft eingesetzt, konnte sich allerdings mittlerweile einen festen Platz in der Robotik sichern. Die besonderen Vorteile dieser Kamera ergeben sich aus dem überaus günstigen Preis im Vergleich zu der sehr guten Qualität der präsentierten Bilder. Darüber hinaus, liefert die Kinect Kamera durch

ihren speziellen Aufbau und den entsprechenden Treibern, eine 3D-Punktwolke und ein Farb-Tiefenbild. Diese beiden Darstellungsformen sind die essenziellen Komponenten in den beiden Versionen des Gesamtsystems.
Um eine Person nun in einem Bild zu detektieren und auch wiederzufinden, müssen verschiedene Funktionen und Algorithmen angewendet werden. Die Kombination aus diesen ergibt schließlich das Gesamtsystem dieser Masterarbeit. Aus diesem Grund mussten nun die verschiedenen Methoden der 2 Dimensionalen-Bildbearbeitung betrachtet werden. Hierbei wurde im Vorfeld ein kurzer Exkurs über die beiden Farbmodelle RGB und HSV erörtert um schließlich diese Methoden zu betrachten, welche für eine erfolgreiche Detektion und Verfolgung einer Person am sinnvollsten und effektivsten waren. Dabei konnten einige Funktionen sehr gute Ergebnisse erzielen und wiederum andere, in diesem speziellen Anwendungsfall, nicht eingesetzt werden. So führte z.B. die Verwendung der FitEllipse-Funktion zur Detektion der Beine einer Person, zu unregelmäßigen Ergebnissen. Die Verwendung des Histogrammvergleichs hingegen, führte zu weitaus regelmäßigeren und besseren Ergebnissen und schließlich auch zur Implementierung in die Version 2 des Gesamtsystems. Im Zuge der Methodenbetrachtung wurde auch der FastTemplateMatcher als mögliches Tool zur Personenverfolgung evaluiert. Dieser konnte schließlich, in abgeänderter Form, in die Version 1 des Gesamtsystems implementiert werden.
Ähnlich der Betrachtung der 2D-Bildbearbeitung, wurden in dem Kapitel der 3D-Bildbearbeitung verschiedene Methoden zur Verwendung und Manipulation von 3D-Punktwolken und Tiefenbildern betrachtet. Dem voran, diente eine Übersicht von erweiterten Grundlagen zum besseren Verständnis im Umgang mit ROS und der theoretischen Tiefenbilderstellung.
Um die Personenverfolgung nochmals stabiler und effektiver zu gestalten wurde eine Funktion ausprogrammiert, welche sich Particle Filter nennt. Dieser verhält sich ähnlich dem Kalman-Filter zur Positionsvorhersage eines Objektes. In diesem Fall diente der Particle Filter zur Abschätzung und Berechnung der aktuellen Position einer Zielperson auf Basis von verschiedenen Funktionen, welche eine Gesamtübereinstimmung zu einem Initialisierungsstatus errechneten. Basierend darauf, wurde der Particle Filter in die Version 2 des Gesamtsystems implementiert.
Neben der Identifikation der Zielperson durch die Kamerabilder, sollte eine weitere Form der eindeutigen Identifizierung implementiert werden. Dazu sollte die Person eine Art Chip oder eine kleine Karte erhalten, welche sie von anderen Personen eindeutig unterscheidet. Da es ohne ein umfassendes Sensornetzwerk nicht möglich ist Positionsinformationen zu errechnen, beschränkte sich die Identifikation der Person lediglich auf die Ermittlung ihrer Distanz. Diese Information sollte schließlich als weiteres Übereinstimmungskriterium in den Particle Filter miteinfließen. Um dies zu verwirklichen wurden verschiedene drahtlose Systeme betrachtet. Diese hatten alle verschiedene Anwendungsgebiete und dementsprechend auch äußerst verschiedene Preisklassen. Aufgrund des knappen Budgets und des Zeitmangels, wurde schließlich das kostengünstige und sofort verfügbare

Bluetooth-System verwendet. Verschiedene Tests ergaben allerdings, dass eine stabile Distanzermittlung nicht möglich war. Die verwendete Bluetooth-Hardware regelte dabei die drei betrachteten Parameter Link Quality Indicator (LQI), Received Signal Strength Indicator (RSSI), und Transmit Power Level (TPL) immer so aus, das eine optimale Sendeleistung gewährleistet wurde. Lediglich über den TPL-Wert konnten einigermaßen Rückschlüsse auf die Distanz zur Person geschlossen werden. Hinzu kam die Trägheit des System und der daraus resultierenden langsamen Werteänderung. Eine sich bewegende Person konnte somit nicht eindeutig verfolgt werden. Ein weiterer Test mit einer aktiven Datenübertragung lieferte hingegen bessere Resultate. Die betrachteten Parameter veränderten sich nun um ein vielfaches schneller und die Distanzveränderung konnte eindeutiger ermittelt werden. Diese konstante Datenübertragung geschah allerdings sehr zu kosten der Akkuleistung, weswegen ein andauernder praktischer Einsatz nicht möglich war. Diese Testresultate führten schließlich zu der Erkenntnis, dass eine andere Bluetooth-Hardware, welche abstimmbarer und besser konfigurierbar ist, durchaus in der Lage wäre zur effektiven Distanzbestimmung der Zielperson beizutragen.

Im weiteren Verlauf wurde schließlich auch die Steuerung des verwendeten Pioneer-Roboters betrachtet. Hierbei wurden die Möglichkeiten der Robotersteuerung über spezielle ROS-Befehle erörtert. Dabei konnte wahlweise auch eine Robotersimulation verwendet werden, sollte der physikalische Roboter im Moment nicht zur Verfügung stehen. Außerdem wurde die Implementierung der verschiedenen Sensoren in Betracht gezogen wobei die Bumper aufgrund eines Wechsels des mobilen Roboters während der Projektlaufzeit nur theoretisch veranschaulicht. Auch der Einsatz des Hokuyo-Laserscanners zur Kollisionsvermeidung und Navigation wurde nur in verschiedenen Tests betrachtet. Aufgrund der begrenzten Projektzeit konnte allerdings keine ausgereifte Funktionalität erzielt werden, was einen Einsatz in das Gesamtsystem nicht möglich machte.

Die abschließenden Kapitel befassen sich im weiteren Verlauf mit den beiden Versionen des Gesamtsystems und der durchgeführten Experimente mit darauffolgender Evaluation dieser. Aus der Kombination der 3D-Punktwolke der Kinect Kamera und dem FastTemplateMatcher entwickelte sich zu Beginn der Projektlaufzeit die Version 1 des Gesamtsystems. In dieser wurden weitere Funktionen kombiniert, welche eine relativ gute Personenverfolgung ermöglichten. Nach einigen Filterfunktionen, welche die 3D-Punktwolke auf eine definierte Größe formte, wurden in der Initialisierungsphase 20 Screenshots von der Zielperson gemacht. Diese musste sich dabei einmal im Kreis drehen, um eine 360°-Erfassung zu ermöglichen. Diese Screenshots halfen im Anschluss in der Programmablaufphase, dem FastTemplateMatcher zur besseren Detektion und Verfolgung der Zielperson. Das Problem, welches hierbei allerdings schnell deutlich wurde, waren die äußerst zeit- und rechenintensiven Operationen zur Erstellung der 3D-Punktwolke und dem Vergleich der Vorlage mit dem aktuellen Bild durch den FastTemplateMatcher. Konnte dieser ein aktuelles Bild nicht eindeutig einer Vorlage

zuordnen, verlangsamte sich die Bearbeitung um ein vielfaches, bis die Geschwindigkeit schließlich von ca. 15 Bilder/s auf 1-2 Bilder/s herabsank. In dieser Zeit blieb der Roboter aufgrund seiner Programmierung automatisch stehen und die Person bewegte sich aus dem sichtbaren Bildbereich hinaus.
Um dieses Zeitproblem zu bewältigen wurde eine Kombination aus dem Tiefenbild der Kinect Kamera und dem Particle Filter mitsamt verschiedener OpenCv Funktionen entwickelt. Allein durch die Verwendung des Tiefenbildes steigerte sich die Geschwindigkeit des Systems auf ca. 25 Bilder/s und ergab somit eine deutliche Beschleunigung der Rechenzeit. Zu Beginn des Programmes wurde ein Initialisierungshistogramm der Zielperson erstellt. Das betrachtete Bild wurde im Vorfeld durch bestimmte Filterfunktionen auf einen definierten Bereich getrimmt, um nur die Farbinformationen der Zielperson im Histogramm zu erhalten. Außerdem wurde eine Initialisierungsdistanz ermittelt welche die aktuelle Position der Zielperson definierte. Diese beiden Parameter wurden schließlich dem Particle Filter übergeben und dienten fortan als Vorlagen bei der Erstellung der jeweiligen Übereinstimmungsergebnissen. Der Particle Filter nutze die Ergebnisse der Gesamtübereinstimmung und konnte dadurch die Person überaus erfolgreich im Bild verfolgen.
Nach der Aneignung der theoretischen Grundlagen und der verschiedenen Implementierungsversionen, konnten verschiedene Testdurchläufe durchgeführt werden. Dabei zeigten sich vor allem die Schwächen der Version 1 des Gesamtsystems im direkten Vergleich zur Version 2 des Gesamtsystems. Bei den ersten Testläufen mit der Version 1 des Gesamtsystems wurde eine Person mit einer langen blauen Jeanshose erfasst und verfolgt. Hierbei konnten relativ gute Ergebnisse erzielt werden, wenngleich die Detektion zeitweise äußerst langsam stattfand. Solange die Person korrekt erfasst wurde, arbeitete das System mit einer moderaten Geschwindigkeit von ca. 15 Bildern/s. Wurde die Zielperson allerdings nicht mehr richtige erkannt, so fiel die Geschwindigkeit des Systems auf 1-2 Bildern/s, da der FastTemplateMatcher die Person nicht mehr eindeutig identifizieren konnte und nun in den anderen Vorlagen nach einer entsprechenden Übereinstimmung suchte. Die Programmierung des Roboters veranlasste ihn zum sofortigen Stillstand. Dies führte im weiteren Verlauf dazu, dass sich die Person außerhalb des Blickfelds der Kamera bewegte und der FastTemplateMatcher schließlich die Person komplett verlor.
Diese Probleme sollten in der Version 2 des Gesamtsystems vermieden werden. Durch die Kombination des Tiefenbildes mit dem Particle Filter konnte ein deutlicher Geschwindigkeitszuwachs erzielt werden. Dies zeigte sich schließlich in den durchgeführten Versuchen. Dabei wurde jeweils die Verfolgung einer Person mit einer langen blauen Jeanshose, einem bunten Rock und einer kurzen blauen Jeanshose getestet. Die Ergebnisse welche sich darauf ergaben waren durchaus vielversprechend. So konnten die Personen relativ gut detektiert und verfolgt werden. Auch bei einer erschwerten Testbedingung durch eine zweite Person im Bild, wurde die Zielperson

weiterhin sicher ermittelt. Probleme bei der internen Farbeinstellung der Kinect Kamera führten zeitweise zu falschfarbigen Bildern, und schließlich zu fälschlichen Übereinstimmungen mit anderen Objekten wie Schränken oder Kartons.
Zusammengefasst betrachtet wurde das Projekt erfolgreich abgeschlossen. Zwar konnten nicht alle optionalen Ziele erfolgreich erreicht werden, das Basissystem konnte allerdings den gestellten Primärzielen durchaus gerecht werden. So wurde eine Person erfolgreich in einem Bild detektiert und schließlich verfolgt. Auch wenn teilweise andere Objekte als korrekte Ziele erkannt wurden, fand eine stabile Detektion und Verfolgung der Zielperson bei konstanten Lichtverhältnissen statt.

Ausblick auf die Zukunft
Das Potential für Erweiterungen und Verbesserungen ist dabei überaus groß. So lässt die Version 2 des Gesamtsystems sehr viel Spielraum für neue Detektionsalgorithmen und besseren Personenerkennung. Neuer Softwarelösungen führen zu einer weitaus besseren Segmentierung der Zielperson und somit auch zu einer besseren Unterscheidung zu anderen Objekten mit nicht-menschlichen Formen, wie Schränke oder Kartons.
Vor allem die Implementierung einer besseren drahtlosen Identifikationsmethode sollte ein Hauptbestandteil in Folgeprojekte sein. Dazu müssen eingehendere Untersuchungen der bereits erwähnten drahtlosen Systeme erfolgen und mehr Hardwarekomponenten getestet werden. Daraus kann nun eine weitaus bessere Evaluierung der Systeme erfolgen, als es in dieser Masterarbeit möglich war. Dabei sollten eine bessere konfigurierbare Bluetooth-Hardware und ein kostengünstiges ZigBee-System einem genaueren Vergleich unterzogen werden, da diese Systeme durchaus ein großes Identifikationspotential bei einem relativ günstigen Kosteneinsatz haben.

Personen wiederfinden (Erweiterungsmöglichkeiten)
Hierbei handelt es sich um eine Erweiterungsmöglichkeit bei einer Verwendung der drahtlosen Systeme. Hat der Roboter einmal seine Zielperson verloren und befindet sich diese außerhalb des Kamera Blickfeldes, können folgende Überlegungen getroffen werden, um diese auch eindeutig im Supermarkt wiederzufinden.

- Beim Registrieren für den neuen Roboter wird die Person (spezielle Merkmale) mittels einer Überwachungskamera gespeichert. Diese Merkmale können nun im gesamten Supermarkt wiedergefunden werden. Vorausgesetzt ist dabei eine durchgehende Abdeckung durch die Überwachungskameras.
- An verschiedenen Positionen im Supermarkt, befinden sich in der Wand oder an den Regalen z.B. eingelassene RFID-Leser. Geht die Zielperson nun an diesen Bereichen vorbei, kann der Weg nachverfolgt.

Die Informationen können nun an den Roboter übermittelt werden und er kann zu seiner Zielperson fahren.

Folgt der Roboter nun einer Zielperson stabil durch den Supermarkt, so kann er auf Wünsche oder Suchanfragen dieser eingehen. Mit einer gespeicherten Inventar- und dazugehörigen Positionsliste, kann der Roboter nun der Person vorfahren und sie damit zum gewünschten Produkt dirigieren. Wird zudem das Fahrgestell des Roboters auf einen Außeneinsatz erweitert (z.B. Kettenantrieb), so kann er die Waren der Person auch bis zum Auto oder sogar bis nach Hause tragen. Dies würde dann möglich sein, wenn der Roboter kommerziell erhältlich wäre und die Person somit einen persönlichen Einkaufsroboter besitzt.

Abbildungsverzeichnis

Abbildung 1: Offizielle Website des RobotaS-Projekts 5
Abbildung 2: Offizielles Logo des RobotaS-Projekts 5
Abbildung 3: Verwendeter Pioneer Roboter 10
Abbildung 4: Die Kinect Kamera mit ihrem kompletten Aufbau 12
Abbildung 5: Ansichten des Hokuyo Laser Scanner 13
Abbildung 6: Sensoraufbau des alten Roboters 14
Abbildung 7: Bild vom verwendeten Controller 15
Abbildung 8: Ansicht in RGB-Farbmodell 17
Abbildung 9: Ansicht in HSV-Farbmodell 17
Abbildung 10: Kantendetektoren im Überblick 19
Abbildung 11: FindContour-Algorithmus auf ein Bild angewendet 20
Abbildung 12: Bewegungserkennung mit Hintergrundentfernung 20
Abbildung 13: Detektion der Beine mit dem FitEllipsen-Algorithmus (oben mit und unten ohne Segmentierung) 21
Abbildung 14: Originalbild mit markiertem (links) und ausgeschnittenem (rechts) Region-of-Interest 23
Abbildung 15: Vorlage (links) und gefundenes Übereinstimmung (Kasten rechts) 24
Abbildung 16: Farbhistogramm aus einem RGB-Bild 26
Abbildung 17: Farbhistogramm aus einem HSV-Bild 26
Abbildung 18: Histogrammvergleich mit der CompareHist-Funktion 28
Abbildung 19: Color Histogram Matching auf ein aktuelles Bild angewandt, ... 29
Abbildung 20: Aufnahmen mit der FloodFill-Funktion 30
Abbildung 21: Der SIFT-Algorithmus in der Praxis 31
Abbildung 22: Schematische Darstellung einer Stereokameraanordnung 36
Abbildung 23: Beispiel eines ROS-Programms in Knotenansicht 38
Abbildung 24: Ausgabe eines Bildes über einen eigenen Knoten 39
Abbildung 25: Beispiele der 3D-Aufnahme 39
Abbildung 26: Ansichten der Seiten- (links) und Abstandsfilterung (rechts) 40
Abbildung 27: Ansichten von der Filterung des Bodens (links) und des Tisches (rechts) .. 41
Abbildung 28: Gefiltertes Tiefenbild 42
Abbildung 29: Gefiltertes Farbbild 43
Abbildung 30: Tiefeninformationen des Bildes farblich nach Abstand dargestellt 43
Abbildung 31: Ansichten der Particles auf einem erfassten Bild 45
Abbildung 32: Abhängigkeit des LQ im Vergleich zur Distanz beim 1sten Versuch 55
Abbildung 33: Abhängigkeit des TPL im Vergleich zur Distanz beim 1sten Versuch 56
Abbildung 34: Abhängigkeit des RSSI im Vergleich zur Distanz beim 1sten Versuch 56
Abbildung 35: Abhängigkeit des LQ im Vergleich zur Distanz beim 2sten Versuch 57
Abbildung 36: Abhängigkeit des TPL im Vergleich zur Distanz beim 2sten Versuch 57

Abbildung 37: Abhängigkeit des RSSI im Vergleich zur Distanz beim 2sten Versuch 58
Abbildung 38: Detektierte Person (links) und Simulation des Roboters (rechts) 66
Abbildung 39: Erkennungsradius des Hokuyo Laserscanners 67
Abbildung 40: Aufbau des Prototypens 69
Abbildung 41: Ablaufdiagramm der Version 1 des Gesamtsystems 70
Abbildung 42: Beispiele der Initialisierungsscreenshots aus verschiedenen Perspektiven 72
Abbildung 43:Umwandlung einer 3D-Punktwolke (links) in ein 2D-Bild (rechts) 75
Abbildung 44: Sicherheits- (links, blauer Kasten) und Initialisierungsschalter (rechts, roter Kasten) des Controllers 77
Abbildung 45: Ablaufdiagramm der zweiten Version des Hauptprogramms 78
Abbildung 46: Ausgabe der reinen Bilddaten, links, RGB-Bild; rechts, Tiefenbild 79
Abbildung 47: Gefilterte Bilder der Kamera, RGB-Bild (links) und Tiefenbild (rechts) 81
Abbildung 48: Erfasste Person mit der Floodfill-Funktion 81
Abbildung 49: Ergebnisse der Histogramme, währen der Initialisierungs- und der Runphase 83
Abbildung 50: Testlauf der Version 2 des Gesamtsystems, einer Person mit einer langen blauen Jeanshose 87
Abbildung 51: Testlauf der Version 2 des Gesamtsystems einer Person mit einer kurzen blauen Jeans 89
Abbildung 52: Korrekte Detektion der Zielperson mit kurzen blauen Jeans (links), trotz erschwerten Bedingungen der Testumstände durch eine weitere Person im Bild (rechts) 89
Abbildung 53: Testlauf der Version 2 des Gesamtsystems einer Person mit einem Rock 90
Abbildung 54: Korrekte Detektion der Zielperson mit Rock, trotz erschwerten Bedingungen der Testumstände durch eine weitere Person im Bild (links und rechts) 90

Tabellenverzeichnis

Tabelle 1: Vor- und Nachteile der Kinect Kamera 12
Tabelle 2: Vor- und Nachteile des Hokuyo Lasersensors 13
Tabelle 3: Vor- und Nachteile der Bumper 14
Tabelle 4: Vor- und Nachteile der Ultraschallsensoren 14
Tabelle 5: CompareHist-Methoden, ihre dazugehörigen Formeln und Wertebereiche 27
Tabelle 6: Zusammenfassung aller betrachteten Methoden der 2-Dimensionalen Bildbearbeitung 33
Tabelle 7: Zusammenfassung aller betrachteten Methoden der 3-Dimensionalen Bildbearbeitung 44
Tabelle 8: Vergleich der Haupt-Funknetzwerke 49
Tabelle 9: Vorteile und Nachteile von RFID-Systemen 50
Tabelle 10: Vorteile und Nachteile von Bluetooth-Systemen 53
Tabelle 11: Vor- und Nachteile des WLAN-Systems 59
Tabelle 12: Vor- und Nachteile des ZigBee-Systems 60
Tabelle 13: Vergleich zwischen ZigBee und Bluetooth 62
Tabelle 14: Preisvergleich zwischen Bluetooth- und RFID-Komponenten 63
Tabelle 15: Größenvergleich zwischen Bluetooth-Transponder und aktiven, passiven RFID-Transpondern 64

Literaturverzeichnis

Bradski, G., Kaehler, A., 2008. Learning OpenCV - Computer Vision with the OpenCV Library. Köln: O'Reilly Media, Inc.

Cytron Technologies, 2009. Mobile Robot Platform [online]. Verfügbar unter: http://tutorial.cytron.com.my/2009/05/26/mobile-robot-platform/ [Zugang am 12. August.2011].

Dembowski, K., 2009. Lokale Netze: Handbuch der kompletten Netzwerktechnik. München: Pearson Deutschland GmbH Verlag.

Farahani, S., 2008. ZigBee wireless networks and transceivers. Frankfurt: Elsevier Information Systems GmbH Verlag.

Finger, M., 2008. Konzipierung eines Identifikationssystems auf Bluetooth-Basis. Dresden: Jörg Vogt Verlag.

Finkezeller, K., 2008. RFID-Handbuch: Grundlagen und praktische Anwendungen von Transpondern, kontaktlosen Chipkarten und NFC. 5.Auflage. München: Carl Hanser Verlag & CO. KG.

Franke, W., 2006. RFID: Leitfaden für die Logistik. Wiesbaden: Gabler Verlag.

Georgiou, T., 2011. Fast Match Template [online]. Verfügbar unter: http://opencv.willowgarage.com/wiki/FastMatchTemplate [Zugang am 12. August.2011].

Gessler, R., Krause, T., 2009. Wireless-Netzwerke für den Nahbereich: Grundlagen, Verfahren, Vergleich, Entwicklung. Wiesbaden: Vieweg +Teubner Verlag.

Grossmann R., Blumenthal, J., Golatowski, F., Timmermann, D., 2007. Localization in Zigbee-based wireless sensor networks [online]. Verfügbar unter: http://www.loms-itea.org/publications/LocalizationZigbee.pdf [Zugang am 12. August.2011].

Hesse, S., Schnell, G., 2009. Sensoren für die Prozess- und Fabrikautomation. 4. Auflage. Wiesbaden: Vieweg+Teubner Verlag.

Hornung, G., 2009. GIMP 2.6: Praxisbuch mit Workshops und Videotutorials. Heidelberg: Hüthig Jehle Rehm Verlag.

Jähne, B., 2005. Digitale Bildverarbeitung. 6.Auflage. Heidelberg: Springer Verlag.

Linuxcommand, 2002. HciTool [online]. Verfügbar unter: http://linuxcommand.org/man_pages/hcitool1.html [Zugang am 12. August.2011].

Linzner, S., 2010. Multimodale Indoor-Lokalisierung für Android basierte mobile Endgeräte [online]. Verfügbar unter: https://ics2.cs.uni-tuebingen.de/publications/papers/linzner_diplomarbeit.pdf [Zugang am 12. August.2011].

Maschke, T., 2004. Digitale Bildbearbeitung. Heidelberg: Springer Verlag.

Neumann, B., 2004. Bildverarbeitung für Einsteiger: Programmbeispiele mit Mathcad. Heidelberg: Springer Verlag.

OpenCv, 2011. Camera calibration and 3d reconstruction [online]. Verfügbar unter: http://opencv.willowgarage.com/documentation/camera_calibration_and_3d_reconstruction.html [Zugang am 12.August.2011].

Sorex, 2008. Wireless Technologie – ZigBee [online]. Verfügbar unter: http://www.sorex-austria.com/zigbee.html [Zugang am 12. August.2011].

University of New South Wales, 2008. Multi-sensor human tracking (final report) [online]. Verfügbar unter: http://cgi.cse.unsw.edu.au/~cs4411/wiki/index.php?title=Multi-sensor_human_tracking_(final_report) [Zugang am 12. August.2011].